# Script description for „cont-3D-snGFR” – 2020/09/15

The following script description gives a detailed outline of the scripts included in this repository, to help understand the different program parts. Minor changes in the associated script/macro might not be updated into this description, however the overall principle of the workflow will not generally differ.

## R-Script

After setting the working directory, the location of the script itself is detected (the ImageJ macro should be located in the same folder). If not already available, the additional library “ggplot2” is installed and loaded. A theme for the general appearance of all graphs is defined. The location of ImageJ is set.

setwd(choose.dir(caption="Directory with .lif-files", default=getwd()))  
script\_location<-dirname(sys.frame(1)$ofile)   
  
inst\_pack<-data.frame(installed.packages())$Package  
# ggplot2 -----------------------------------------------------------------  
if("ggplot2"%in%inst\_pack){  
 library(ggplot2)  
}else{  
 warning("Package \"ggplot2\" is not installed")  
 install.packages("ggplot2")  
 library(ggplot2)  
}  
  
mytheme3 <- theme(legend.text = element\_text(size = 15),   
 axis.title = element\_text(size = 15),  
 plot.subtitle = element\_text(size = 15,hjust=0.5),  
 axis.text = element\_text(size = 13),   
 axis.line = element\_line(size = 1,colour = "black"),   
 axis.ticks = element\_line(colour="black",size = rel(3)),  
 panel.background = element\_rect(fill = "white",  
 colour="black"),   
 legend.key = element\_rect(fill = "white"),

panel.grid.major = element\_line(colour="grey"),  
 panel.grid.minor = element\_blank(),  
 legend.background = element\_rect(fill="white",  
 colour="black"),  
 legend.title = element\_text(face = "bold", size = 15),   
 plot.title = element\_text(face = "bold",  
 size = 18,hjust=0.5),  
 strip.text.x = element\_text(size = 15))

After setting up these initial parameters, the ImageJ macro is called from the command line and all .lif – files in the previously set working directory are processed.

# ImageJ Macro*------------------------------------------------*

dir\_fiji<-**choose**.**files**(caption="Select executable FIJI file")

system(paste0(dir\_fiji," -macro ",script\_location, "/cont-3D-snGFR.ijm ",

**getwd**()))

Once the image processing is done, and all result tables are saved, they are further analyzed in the R-Script, resulting in one summary table with the calculated snGFR, R-square of the regression line, total volume and length of the measured proximal tubule and the number of datapoints used for linear regression (“dfsum”).

# R Script ----------------------------------------------------------------  
if(!dir.exists(paste0("Graphs\_", Sys.Date()))){  
 dir.create(paste0("Graphs\_", Sys.Date()))  
}  
  
framerate<-as.numeric(readline(prompt="Framerate in time series [fps]: "))

filelist<-list.files("Results", pattern=".txt")  
dfsum<-c("Name", "Median slope", "Slope/GFR", "R", "Volume",  
 "Length","Datapoints")  
for(a in 1:length(filelist)){  
 if(length(grep("Volume", filelist[a]))!=1){  
 if(length(grep("Settings", filelist[a]))!=1){  
 #data\_process(filelist[a])  
 }  
 }  
}  
write.table(dfsum, paste0(Sys.Date(),"-Result\_summary.txt"), sep="\t",  
 row.names=F, col.names=F)

During this data analysis, the result tables from the intensity measurements in the time series (“df1”) and the z-stack (“dfvolume”) are loaded. The volume (dfvolume$volume) is interpolated for every position along the line ROI for the intensity measurement (df1$length), visualized and saved.

df1<-read.table(paste0("Results/", filename), header=T, sep="\t")

nframes<-max(df1$frame) # number of frames  
df2<-matrix(NA, ncol=nframes, nrow=nrow(df1)/nframes)  
   
colnames(df2)<-1:nframes  
rownames(df2)<-df1$length[1:nrow(df2)]  
   
# volume over length of tubulus-----  
dfvolume<-read.table(paste0("Results/Volume\_",filename),  
 header=T, sep="\t") #volume measurement  
  
PT\_length<-df1$length[df1$frame==1]  
PT\_volume<-approx(dfvolume$Volume, n=length(PT\_length))$y  
PT\_difvolume<-c(0,diff(PT\_volume))  
   
df1$volume<-rep(PT\_volume, times=nframes)  
df1$difvolume<-rep(PT\_difvolume, times=nframes)  
   
# crop table to where volume rendering vs. position is increasing  
df2<-df1[df1$frame==1,]  
df1<-subset(df1, difvolume>0.5\*mean(df2$difvolume))  
df2<-df1[df1$frame==1,]  
   
# plot volume against position  
p<-ggplot(df2, aes(x=length, y=volume))+  
 geom\_point()+  
 mytheme3+  
 labs(title="PT Volume against length",  
 subtitle=filename, x="[?m]", y="[?m?]")  
   
print(p)  
ggsave(paste0("Graphs\_", Sys.Date(),"/Pos+Vol\_", filename, ".png"),  
 width=15, height=10, unit="cm")

Afterwards, the intensities measured in the time series are smoothed, to reduce the effects of signal noise and the intensity changes over time are calculated as the differential of intensity.

# smooth intensity  
realpos<-df1$volume[df1$frame==1]  
maxdifintens<-NA  
maxdifslope<-NA  
x1<-0  
for(x in realpos){  
 x1<-x1+1  
 df1$intensity[df1$volume==x]<-smooth.spline(df1$intensity  
 [df1$volume==x],  
 spar=0.5)$y  
 df1$difintens[df1$volume==x]<-c(NA, diff(df1$intensity[df1$volume==x]))  
 maxdifintens[x1]<-df1$intensity[df1$volume==x][which(diff(df1$intensity[df1$volume==x]==max(diff(df1$intensity[df1$volume==x])))]

maxdifslope[x1]<-df1$difintens[df1$volume==x][which(diff(df1$intensity[df1$volume==x])==max(diff(df1$intensity[df1$volume==x])))]

}

Subsequently, different statistics for changes of intensity over time are calculated, to define the frames of interest (frames with general changes in signal intensity, up to maximum intensity, before intensity decreases again) and the threshold intensity (intensity with the maximum median slope across all frames – estimated turning point of the curve).

# plot max intensities per frame and plot max intensity curve -------------  
dfstat<-aggregate(df1$intensity, by=list(df1$frame), FUN="mean")  
dfstat$std<-aggregate(df1$intensity, by=list(df1$frame), FUN="sd")$x  
dfstat$max<-aggregate(df1$intensity, by=list(df1$frame), FUN="max")$x  
dfstat$min<-aggregate(df1$intensity, by=list(df1$frame), FUN="min")$x  
dfstat$max<-smooth(dfstat$max)  
dfstat$diffmax<-c(0, diff(dfstat$max))  
maxdiffmax<-max(dfstat$diffmax) # maximum slope of intensity changes  
min\_frame<-min(dfstat$Group.1[dfstat$diffmax>=0.5\*maxdiffmax])-5  
max\_frame<-max(dfstat$Group.1[dfstat$max>=0.8\*max(dfstat$max)])+5  
TS\_intens<-median(maxdifintens) # ts intensity in turning point of curve   
p<-ggplot(dfstat, aes(x=Group.1, y=max))+  
 geom\_jitter()+  
 expand\_limits(y=0)+  
 labs(subtitle=filename, title="Maximum intensity per frame",  
 y="intensity", x="frame")+  
 mytheme3+  
 geom\_hline(yintercept=TS\_intens)+  
 scale\_y\_continuous()  
print(p)  
ggsave(paste0("Graphs\_", Sys.Date(),"/Maximum\_", filename, ".png"),  
 width=15, height=10, unit="cm")

For visualization, the original dataframe is reduced to only include the intensity changes at 20 different positions along the proximal tubulus.

# reduce dataframe for printing (print only 20 positions)---------------  
realpos<-df2$volume[df2$frame==min\_frame]  
df3<-df2  
nopos<-length(df2$volume[df2$frame==min\_frame]) # number of positions  
facpos<-floor(nopos/20) # which positions to take  
if(nopos>20){  
 realpos<-df2$volume[df2$frame==min\_frame][(1:20)\*floor(nopos/20)]  
}else{  
 realpos<-realpos  
}  
   
df3<-subset(df3, df3$volume%in%realpos)  
  
p<-ggplot(df3, aes(x=frame, y=intensity, fill=volume,  
 colour=volume, group=volume))+  
 scale\_colour\_gradient2(midpoint=midx, low="blue", mid="gray",  
 high="red", space ="Lab" )+  
 scale\_fill\_gradient2(midpoint=midx, low="blue", mid="gray",  
 high="red", space ="Lab" )+  
 geom\_line(size=1)+  
 labs(title="Intensity changes over time for every position",  
 subtitle=filename, y="Intensity", x="Frame", colour="Volume")+  
 mytheme3+  
 geom\_hline(yintercept=as.numeric(TS\_intens), size=2, colour="green")+  
 expand\_limits(y=0)+ scale\_x\_continuous(limits=c(min\_frame, max\_frame))+  
 geom\_vline(xintercept = min\_frame:max\_frame, colour="black")  
   
print(p)  
ggsave(paste0("Graphs\_", Sys.Date(),"/Cleaner\_Frames\_", filename,  
 ".png"), width=20, height=15, unit="cm")

Then, the dataset is further reduced, to only include values up to the global maximum of the intensity curves.

# reduce to max intensity/position ("wave")-------------------  
realpos<-df2$volume[df2$frame==1]  
df3<-df2  
for(x in realpos){  
 dfx<-subset(df3, volume==x)  
 fmax<-dfx$frame[which(dfx$intensity==max(dfx$intensity))[1]]  
 if(fmax<max(df2$frame)){  
 df3<-df3[-which(df3$volume==x&df3$frame>fmax),]  
 }   
}  
df2<-df3

With this datatable, the intercept of every position (meaning, the cumulative volume of the PT at this position) with the threshold intensity is approximated.

# intercept of position with frame at ts-intensity----------------  
dfx<-data.frame("frame"=min\_frame:max\_frame)  
dfx$intensity<-NA  
dfx$length<-NA  
dfx$volume<-NA  
dfx$difintens<-NA  
   
for(x in 1:nrow(dfx)){  
 vx<-dfx$frame[x]  
 dfy<-subset(df2,frame==vx)   
 if(nrow(dfy)>1){  
 fitlength<-loess(intensity~length, data=dfy)  
 fitvolume<-loess(length~volume, data=dfy)  
 fitdifintens<-loess(intensity~difintens, data=dfy)   
 dfx$length[x]<-approx(x=dfy$intensity, y=dfy$length,  
 xout=TS\_intens)$y  
 dfx$volume[x]<-approx(x=dfy$intensity, y=dfy$volume,  
 xout=TS\_intens)$y   
 dfx$intensity[x]<-TS\_intens  
 dfx$difintens[x]<-approx(x=dfy$intensity, y=dfy$difintens,  
 xout=TS\_intens)$y  
 }  
}   
dfx<-na.omit(dfx)

The units in this dataframe are then converted (frame to minutes and µm³ to nl), and used for linear regression.

# adjust units ------------   
dfx$frame<-dfx$frame/60/framerate  
dfx$volume<-dfx$volume/1000000

# linear regression ----------------------  
midx<-max(dfx$volume)/2  
m <- lm(volume ~ frame, dfx)  
b = unname(coef(m)[2])  
r2=summary(m)$r.squared  
   
p<-ggplot(dfx, aes(x=frame, y=volume, colour=volume))+  
 geom\_point()+  
 geom\_smooth(method="lm", formula=y~x)+  
 geom\_line()+  
 scale\_colour\_gradient2(midpoint=midx, low="blue", mid="gray",  
 high="red", space ="Lab" )+  
 geom\_text(x = max(dfx$frame, na.rm=T),  
 y = 1.01\*min(dfx$volume, na.rm=T),  
 label = lm\_eqn(dfx), parse = TRUE, size=5, hjust=1,  
 colour="black")+  
 labs(title="snGFR", subtitle=filename, y="[nl]", x="[min]",  
 colour="Volume")+  
 mytheme3  
print(p)  
  
ggsave(paste0("Graphs\_", Sys.Date(),"/Regression\_", filename, ".png"),  
 width=20, height=15, unit="cm")  
dfsum<-rbind(dfsum, c(filename,  
 median(diff(dfx$volume)/diff(dfx$frame)),  
 b, r2, max(dfvolume$Volume),  
 max(df1$length), nrow(dfx)))  
assign("dfsum", dfsum, envir = .GlobalEnv)

## ImageJ Macro

As part of the R-Script, an ImageJ macro is executed to process all .lif-files in the working directory. To open the .lif-files the Bioformat Macro Extensions are loaded and files that have to be processed are listed. Additional code clears the results table and ROI manager and sets options for measurement.

// Working Directory

argument=getArgument();

path=argument+"\\"

// Setup

run("Bio-Formats Macro Extensions");

close("\*");

roiManager("Reset");

roiManager("Show All");

run("Set Measurements...", "standard median area\_fraction redirect=None decimal=6");

run("Clear Results");

// Files

File.makeDirectory(path+"Results");

files=getFileList(path);

For the settings, the macro goes through all .lif-files and the included scenes and successively opens scenes labelled with “LY”. In channel 2 of this time series (LuciferYellow) a median filter (radius 1) is applied and a manual polyline selection has to be made to indicate the direction and position of the flow in the proximal tubulus. This line ROI is spline fit, slice information is removed and it is saved.

selectImage("Current");

run("Duplicate...", "duplicate channels=2-2");

close("Current");

rename("Current");

setLocation(0,0,800,800);

run("Median...", "radius=1 stack");

run("Enhance Contrast...", "saturated=0");

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

setSlice(frames/4\*3);

// line along center of tubule

setTool("polyline");

run("Line Width...", "line=10");

waitForUser("Draw a line along the center of the tubule");

while(selectionType()==-1){

waitForUser("Draw a line along the center of the tubule");

}

run("Fit Spline");

roiManager("Add");

roiManager("Select", 0);

roiManager("remove slice info");

Afterwards, a dialog is presented to indicate which z-stack corresponds to the previous time series.

//Manual selection corresponding z-stack

Dialog.create("Select corresponding z-stack");

Dialog.addMessage("For sample "+seriesName);

serieslist=newArray(1);

for(c=0; c<seriesCount; c++){

Ext.setSeries(c);

Ext.getSeriesName(seriesName);

if(matches(seriesName, ".\*PT.\*")){

serieslist=Array.concat(serieslist, seriesName);

}

}

Dialog.addChoice("Series", serieslist);

Dialog.show();

zseries=Dialog.getChoice();

These settings (name of time series and z-stack) are saved. Once the selections have been made for all scenes in all .lif-files, they are used for measurements. At first, the time series and corresponding line ROI are opened. A median filter (radius 1) is applied to channel 2 (LuciferYellow). For every frame of the time series a x-y intensity plot is generated along the line ROI and numerical results are saved in the results table.

run("Duplicate...", "duplicate channels=2-2");

close("Current");

rename("Current");

//Preprocess stack-----------------------------------------------------

setLocation(500,20,1000,1000);

run("Median...", "radius=1 stack");

run("Enhance Contrast...", "saturated=0");

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

setSlice(frames/4\*3);

//line along center of tubule

roiManager("Select", 0);

run("Plots...", "width=530 height=300 font=12 draw draw\_ticks auto-close minimum=0 maximum=0 interpolate");

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

run("Clear Results");

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

selectWindow("Current");

setSlice(c);

run("Plot Profile");

Plot.getValues(x, y);

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

setResult("length", nResults, x[i]);

setResult("intensity", nResults-1, y[i]);

setResult("frame", nResults-1, c);

}

close();

}

Channel 4 (Hoechst - Nuclei) is also duplicated and the last frame is later used to find the layer of the time series in the z-stack of the proximal tubulus.

run("Duplicate...", "duplicate channels=4-4");

rename("Nuclei");

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

setSlice(slices);

run("Duplicate...", "slice");

rename("Compare");

close("Nuclei");

For volume measurement the z-stack of the proximal tubulus has to be converted to a binary volume with a series of image processing steps and a 3D watershed. Filters are adapted with parameters as a factor of voxel size in every dimension. After opening the scene of the z-stack corresponding to the previously measured time series, a 3D Median (radius is 3 x voxel size) is applied and the contrast is enhanced (saturation of 1% of the pixels) in channels 2-4. Channel 1 is closed.

//Preprocess stack-----------------------------------------------------

run("Split Channels");

close("C1-Volume");

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

setSlice(slices/1.5);

imglist=getList("image.titles");

getVoxelSize(width, height, depth, unit);

vwidth=width;

vdepth=depth;

for(c=0;c<imglist.length; c++){

selectWindow(imglist[c]);

if(imglist[c]!="Compare"){

setSlice(slices/1.5);

run("Median 3D...", "x="+3\*vwidth+" y="+3\*vwidth+" z="+3\*vdepth);

run("Enhance Contrast", "saturated=1");

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

}

}

To find the slice in the z-stack, that represents the position in which the time series was acquired, the previously generated image of the nuclei (channel 4) from the time series is used. A 3D median filter is also applied to this image (radius is 3 x voxel size) and contrast is enhanced specifically 15 pixel around the line ROI. With the ImageCalculator the difference between channel 4 of the z-stack and the time series is calculated.

selectImage("Compare");

getVoxelSize(width, height, depth, unit);

vwidth=width;

run("Median...", "radius="+3\*vwidth);

roiManager("Select", 0);

run("Line to Area");

run("Enlarge...", "enlarge=15 pixel");

run("Enhance Contrast", "saturated=1");

run("Select None");

run("Apply LUT");

imageCalculator("Difference create stack", "C4-Volume","Compare");

close("Compare");

rename("Compare");

run("Invert", "stack");

For every slice of the resulting 32-bit stack of the difference between the z-stack and the last frame of the time series the mean pixel intensity is measured and compared. The slice with the highest mean pixel intensity (lowest difference) is defined as the target slice within the z-stack: The slice in which the time series was most likely acquired. The previously defined line ROI is updated to be connected to this slice and saved again.

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

xmean=newArray(slices);

xsd=newArray(slices);

xcompare=newArray(slices);

targetslice=1;

finalcompare=0;

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

setSlice(d);

getRawStatistics(nPixels, mean, min, max, std, histogram);

xmean[d-1]=mean;

if(xmean[d-1]>finalcompare){

finalcompare=xmean[d-1];

targetslice=d;

}

}

print(targetslice);

selectWindow("C2-Volume");

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

setSlice(targetslice);

roiManager("Select", 0);

setSlice(targetslice);

roiManager("Update");

Then, channel 4 is processed with additional filters (Variance – radius = 2, 3D Maximum – radius = 4 x voxel size, 3D Median – radius = 4 x voxel size) to exaggerate the signal of second harmonic generation that is visible in this channel. This image is converted to a mask (method = Default) and subtracted from channel 2 (LuciferYellow) after background subtraction (rolling ball = 50). This step is necessary since second harmonic generation signal is also visible in the LuciferYellow channel and would disturb the watershed with false positive signal. Channel 3 is also subtracted from channel 2, to reduce spectral bleed through artifacts caused by the vessel dye.

selectImage("C4-Volume");

getVoxelSize(width, height, depth, unit);

vwidth=width;

vdepth=depth;

run("Variance...", "radius=2 stack");

run("Maximum 3D...", "x="+4\*vwidth+" y="+4\*vwidth+" z="+4\*vdepth);

run("Median 3D...", "x="+4\*vwidth+" y="+4\*vwidth+" z="+4\*vdepth);

run("Convert to Mask", "method=Default background=Dark calculate black");

selectImage("C2-Volume");

run("Subtract Background...", "rolling=50 stack");

imageCalculator("Subtract create stack", "C2-Volume","C3-Volume");

imageCalculator("Subtract create stack", "Result of C2-Volume","C4-Volume");

rename("Volume");

To reduce the image complexity, a distance of 15 pixel around the line ROI is cleared and contrast is enhanced.

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

setSlice(slices/1.5);

roiManager("Select", 0);

run("Line to Area");

run("Enlarge...", "enlarge=15 pixel");

run("Clear Outside", "stack");

run("Enhance Contrast...", "saturated=0");

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

run("Select None");

selectImage("Volume");

The settings for the 3D watershed (threshold value) are obtained from statistics of pixel intensity around the line ROI (mean and standard deviation). For this the line ROI is converted to an area, but only pixels with an intensity > 1 are taken into account for the measurement.

roiManager("Select", 0);

Roi.getPosition(channel, slice, frame);

sliceno=slice;

run("Line to Area");

roiManager("Add");

setThreshold(1, 255);

run("Create Selection");

if(selectionType()==-1){

setBatchMode("exit and display");

exit("Measure intensity in volume measurement: no threshold");

}

roiManager("Add");

roiManager("Select", newArray(1,2));

roiManager("AND");

getStatistics(area, mean, min, max, std, histogram);

All signal with an intensity of 2 x the measured mean intensity around the line ROI is considered to be artifacts and removed.

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

setBackgroundColor(0,0,0);

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

setSlice(c);

setThreshold(2\*mean, 255);

run("Create Selection");

if(selectionType()!=-1){

run("Clear", "slice");

}

}

For the 3D watershed the image is downscaled (x 0.5), to speed up processing and a single pixel in the center of the line ROI is filled (RGB: 255/255/255) as a seed point for 3D watershed. The image threshold is defined as the mean minus standard deviation from the previous measurement of pixel intensity along the line ROI. The resulting binary z-stack is converted to 8-bit and scaled to the original size.

run("Scale...", "x=0.5 y=0.5 z=1.0 interpolation=Bilinear average process create");

rename("Scaled");

close("ZReduced");

selectImage("Scaled");

roiManager("Select", 0);

run("Scale... ", "x=0.5 y=0.5");

getSelectionCoordinates(xpoints, ypoints);

run("Select None");

run("Duplicate...", "duplicate");

setSlice(sliceno);

makeRectangle(xpoints[xpoints.length/2], ypoints[ypoints.length/2], 1,1);

run("Fill", "slice");

rename("peaks");

//Watershed

run("3D Watershed", "seeds\_threshold=1 image\_threshold="+mean-std+" image=Scaled seeds=peaks radius=20");

run("8-bit");

run("Select None");

run("Scale...", "x=2 y=2 z=1.0 interpolation=Bilinear average process create");

The result of the 3D watershed is processed by removing bright (radius = 20) and dark (radius = 30) outliers and applying a median filter (radius = 4 x voxel size).

run("Remove Outliers...", "radius=20 threshold=1 which=Bright stack");

run("Remove Outliers...", "radius=30 threshold=100 which=Dark stack");

run("Median 3D...", "x="+4\*vwidth+" y="+4\*vwidth+" z="+4\*vdepth);

Since it is possible, that in the watershed additional seed points were detected outside the main proximal tubulus, only the largest connected volume in the binary z-stack is used for further analysis. After 3D segmentation (to create a label z-stack) and geometrical measurement, the volumes of the detected particles are compared and the largest volume is detected. Every volume with a different pixel intensity is then cleared.

run("3D Manager Options", "volume distance\_between\_centers=5 distance\_max\_contact=0.9 drawing=Contour");

run("Clear Results");

run("3D Simple Segmentation", "low\_threshold=1 min\_size=0 max\_size=-1");

close("Bin");

close("watershed");

run("3D Geometrical Measure");

max=0;

maxobj=0;

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

V=getResult("Volume(pix)", b);

if(V>max){

max=V;

maxobj=b;

value=getResult("Value", b);

}

}

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

if(max>0){

//erase smaller blobs

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

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

setSlice(c);

setThreshold(value, value);

run("Create Selection");

if(selectionType()!=-1){

setBackgroundColor(0, 0, 0);

run("Clear Outside", "slice");

setForegroundColor(255,255,255);

run("Fill", "slice");

run("Select None");

}else{

run("Select All");

run("Clear", "slice");

}

}

}

The result is then filtered with a Maximum 3D filter (radius = 3 x voxel size) and bright outliers are removed (radius = 10 x pixel size) to smooth the outlines of the segmented object. The original scale is set and the z-stack is saved for visual verification of the segmentation.

run("Maximum 3D...", "x="+3\*vwidth+" y="+3\*vwidth+" z="+3\*vdepth);

run("Remove Outliers...", "radius="+10\*vwidth+" threshold=1 which=Bright stack");

run("8-bit");

run("Properties...", "unit=micron pixel\_width="+vwidth+" pixel\_height="+vwidth+" voxel\_depth="+vdepth);

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

saveAs("TIFF", filepath);

Finally the cumulative volume of the proximal tubule - the segmented object – is measured. For this for every position along the line ROI (for every selection coordinate) a line is created and rotated by 90°. The length of the line is calculated and elongated to 50 x pixel size. After recalculating the position of the two x and y coordinated that define the beginning and endpoint of the elongated line a new line can be created and converted to an area – a perpendicular slice through the z-stack. This area is converted to a selection and added to the ROI manager. For every position along the line ROI, the previously generated “slices” are added up and the object volume within this cumulative selection is measured.

getSelectionCoordinates(x, y);

volume=newArray(x.length-1);

for(c=0; c<x.length-1; c++){

run("Select None");

run("Duplicate...", "duplicate");

makeLine(x[c], y[c], x[c+1], y[c+1]);

run("Rotate...", " angle=90");

getLine(x1, y1, x2, y2, lineWidth);

is\_length=sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2));

delta\_x=(x1-x2)\*50\*vwidth/is\_length;

delta\_y=(y1-y2)\*50\*vwidth/is\_length;

x1\_x=x1-((delta\_x/2)-(x1-x2));

x2\_x=x2+((delta\_x/2)-(x1-x2));

y1\_x=y1-((delta\_y/2)-(y1-y2));

y2\_x=y2+((delta\_y/2)-(y1-y2));

makeLine(x1\_x, y1\_x, x2\_x, y2\_x);

run("Line to Area");

run("Enlarge...", "enlarge=1 pixel");

roiManager("Add");

if(c>0){

roiManager("Select", newArray(roiManager("count")-1, roiManager("count")-2));

roiManager("Combine");

roiManager("Add");

roiManager("Select", roiManager("count")-2);

roiManager("Delete");

}

roiManager("Select", roiManager("count")-1);

run("Clear Outside", "stack");

run("Select None");

run("3D Geometrical Measure");

volumex=0;

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

volumex=volumex+getResult("Volume(unit)", d);

}

volume[c]=volumex;

open\_img=getList("image.titles");

for(d=0; d<open\_img.length; d++){

if(open\_img[d]!="watershed"){

close(open\_img[d]);

}

}

run("Clear Results");

}