**Supplement**

****

*Table 1 – Comparison of volumes obtained with our algorithm (MPV) with that of VFMA. Units of VFMA volumes have been converted from steradians (Sr) to square degrees using the conversion 1Sr = (180/π)2 ~ 3282.8 square degrees.*

S1 - Main Code:

**####################################################################**

**## MAIA Microperimetry volume code - creates a 3D hill of ##**

**## vision with an associated volumetric measure ##**

**## ##**

**## Use of the following code in publications is permitted on ##**

**## condition that a citation of the authors journal submission ##**

**## (pending publication) is provided: ##**

**## ##**

**## "Microperimetry hill of vision and volumetric measures of ##**

**## retinal sensitivity Amandeep Singh Josan, Thomas M W Buckley, ##**

**## Laura J Wood, Jasleen K Jolly, Jasmina Cehajic-Kapetanovic ##**

**## and Robert E MacLaren" ##**

**## ##**

**## please also cite the appropriate ##**

**## packages contained within this code (ggplot2,rgl,fields). ##**

**## This program is intended for research use only. Clinical ##**

**## decisions should not be made based on information generated ##**

**## using this program. ##**

**####################################################################**

**rm(list = ls(all = TRUE))**

**library(ggplot2)**

**library(plyr)**

**library(concaveman)**

**library(tidyverse)**

**library(pracma)**

**library(ggalt)**

**library(dplyr)**

**library(fields)**

**library(plot3D)**

**library(misc3d)**

**library(rgl)**

**library(ggpmisc)**

**library(reshape2)**

**library(openxlsx)**

**library(filenamer)**

**library(data.table)**

**##===========================================================**

**## Insert MAIA file location and filename below**

**##===========================================================**

**setwd("C:/insert path to folder with MAIA threshold.txt file here")**

**#### read from MAIA thredshold.txt output file**

**data0<- read.table("maia-xxxx\_xxx\_xxxx\_threshold.txt", header =T, stringsAsFactors = F, skip=48)**

**#============================================================**

**### =============== Start of program ========================**

**#############################################################**

**data <- data0[,c("ID","x\_deg","y\_deg","Threshold")]**

**colnames(data) <- c("ID", "x","y","thresh")**

**#### remove blindspot test**

**data <- data[!data$ID==0,]**

**#### flip y-coords to match maia output (retinal space as opposed to projected space)**

**data$y <- data$y\*(-1)**

**#### order the rows**

**data <- data[with(data, order((data$y), (data$x))), ]**

**#-------------------------------------------------------------**

**#############################################################**

**############# specify coordinates of interest ###############**

**#############################################################**

 **xmin <- min(data$x)-2**

 **xmax <- max(data$x)+2**

 **ymin <- min(data$y)-2**

 **ymax <- max(data$y)+2**

 **coord <- coord\_cartesian(xlim=c(xmin,xmax),ylim=c(ymin,ymax))**

**#############################################################**

**########### optional segmented analysis #############################**

**### use eccentricity to specify regions**

**#data <- mutate(data, eccen = sqrt((x^2)+(y^2)))**

**#data <- data[data$eccen<90,] # consider total field**

**#data <- data[data$eccen<=8,] # central field only (e.g. central 8 degrees)**

**#data <- data[data$y>0,] # superior field only etc...**

**##############################################################**

**##############################################################**

**## change MAIA definitions of not seen -1db to 0db and seen at**

**## brightest 0db to a small no. <- will alter vol very slightly**

**data$thresh[data$thresh==0] <- 0.1 ## 0.1 is arbitrary but in line with method used by Octopus 900**

**data$thresh[data$thresh==-1] <- 0**

**meanthresh <- mean(data$thresh)**

**meanthresh = formatC(meanthresh, digits = 1, format = "f")**

**### calculated MAIA mean threshold output**

**data1 <- data0[,c("ID","x\_deg","y\_deg","Threshold")]**

**colnames(data1) <- c("ID", "x","y","thresh")**

**data1 <- data1[!data1$ID==0,] ## removes blindspot for mean threshold calc**

**MAIAmeanthresh <- mean(data1$thresh)**

**MAIAmeanthresh[MAIAmeanthresh<0] <- 0 ## if resulting MS is <0 report back as MS=0dB**

**MAIAmeanthresh = formatC(MAIAmeanthresh, digits = 1,**

 **format = "f") ## report to 1 decimal place**

**#############################################################**

**#### colour scheme and legend breaks to approx match MAIA ########**

**#############################################################**

**palette = c("#000000","#4F1B87",**

 **"#A10974","#A30A5E","#B30B43","#D6083C","#FA1B23","#F7131B",**

 **"#FF3037","#F73B3E","#F54A20","#F75128","#F75931","#ED582F",**

 **"#ED5B32","#EB5426","#F56231","#FF722B","#F08827","#FF9C38",**

 **"#FFA442","#FFAB4A","#FFC04A","#FFFF4A","#E1FF4A","#C7FF57",**

 **"#BDFF42","#B3FA2F","#9FFA2F","#76FA2F","#52ED2B","#2ED622",**

 **"#29CC21","#26C720","#24BF1F","#20B51D","#139911","#0B8A0B"**

 **)**

**#-------------------------------------------------------------**

**#-------------------------------------------------------------**

**b = c(0,0.09,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37)**

**lab = c("<0","0","1","2","3","4","5","6","7","8","9","10","11","12",**

 **"13","14","15","16","17","18","19","20","21","22","23",**

 **"24","25","26","27","28","29","30","31","32","33","34",**

 **"35","36")**

**### factorise the thresholds to categorise into legend values**

**data$thresh\_f <- cut(data$thresh, breaks = b, right = FALSE)**

**my\_breaks <- levels(data$thresh\_f)**

**#===========================================================**

**#--------------- pointmap Plot Theme -----------------------**

**theme <- theme\_bw()+theme(**

 **legend.title = element\_text(color = "black", size = 10),**

 **legend.text = element\_text(color = "black", size = 8),**

 **axis.text=element\_text(size=10),**

 **axis.title=element\_text(size=12,face="bold"),**

 **legend.key.size = unit(0.2, "cm"),**

 **axis.line = element\_line(colour = "black"),**

 **legend.key = element\_rect(fill = "white"),**

 **panel.grid.major = element\_blank(),**

 **panel.grid.minor = element\_blank(),**

 **plot.title = element\_text(size=9, hjust=0.95,**

 **margin = margin(b = -10))**

 **)+ theme(aspect.ratio=1)**

**#===========================================================**

**#### create convex hull just for visualisation purpose - Fields does this automatically**

**hull <- data %>%**

 **slice(chull(x, y))**

**######################## PLOTS ##############################**

**#####################################**

**##### print pointmap fields plot**

**#####################################**

**pointmap <- ggplot(data, aes(x=x, y=y, colour=thresh\_f)) +**

 **geom\_point(size=1.5) +**

**# geom\_point(data = hull, col="darkblue", fill=NA)+ ## visualise convex/concave hull points**

 **geom\_polygon(data = hull, col="darkblue", fill=NA)+ ## visualise convex/concave hull line**

 **scale\_colour\_manual(values=palette, lab=lab, drop=F, ## creates legend**

 **name="[dB]")+**

 **geom\_text(aes(label=thresh), hjust=-0.8, vjust=-0.1, ## adds threshold values to pointmap**

 **size=1.9, col="black")+**

 **labs(x = "x (degrees)", y = "y (degrees)")+**

 **guides(col = guide\_legend(override.aes = list(shape = 15,**

 **size = 3.5)))+**

 **theme +**

 **coord**

**print(pointmap)**

**##################################**

**##### generate heatmap and 3D map**

**##################################**

**#########################################################**

**#------------- TPS interpolation below ---------------**

**#----------------------------------------------------------**

**###########################################################**

**############### Interpolate with TPS ######################**

**####### use Thin Plate Spline from Fields package #########**

**nx<-400**

**ny<-400 ## can alter resolution**

**df <- nrow(data1)**

**### main interpolation below**

**tps\_int <- fields::Tps(data.frame(data$x,data$y),**

 **data$thresh, m=2, df=df) ## state number of degrees of freedom (number of threshold tests points)**

 **## GCV performed based on above parameters to provide optimal smoothness of fit**

**tps <- predictSurface(tps\_int, nx=nx, ny=ny) ## predicts data forming 3D surface of HOV**

**### following 3 lines remove possible artifacts arising from interpolation**

**tps$z[tps$z<0] <- 0 ## interpolation with polynomials occasionally cause values to dip below 0dB - clip here to avoid**

**tps$z[tps$z==0] <- 1e-04 ## artifact with black colouring at 0dB. replace with very small figure to get consistent black. Has negligible effect on volume measures**

**tps$z[tps$z>36] <- 36 ## interpolation with polynomials occasionally cause values to dip above 36dB - clip here to avoid**

**#---------------------**

**## rearrange newly interpolated data**

**dftps <- reshape2::melt(tps$z, na.rm = T)**

**names(dftps) <- c("x", "y", "thresh")**

**##### factorize dataframe tps to create breaks to make like MAIA threshold categories**

**dftps$thresh\_ftps <- cut(dftps$thresh, breaks = b, right = F)**

**my\_breaks <- levels(dftps$thresh\_ftps)**

**dftps$x <- tps$x[dftps$x]**

**dftps$y <- tps$y[dftps$y]**

**plot\_tps <- ggplot(dftps, aes(x, y, z = thresh\_ftps)) +**

 **geom\_tile(aes(fill = thresh\_ftps)) +**

 **scale\_fill\_manual(breaks=my\_breaks, values=palette,**

 **labels=lab, drop=F, name="[dB]") +**

 **theme +**

 **coord**

**################################**

**### vol under 3d surface calc**

**################################**

**xlim <- range(tps$x)**

**ylim <- range(tps$y)**

**## the size of each grid cell (a rectangular cell) is:**

**cell\_size <- (diff(xlim)/nx) \* (diff(ylim)/ny)**

**## can convert units by changing z below (e.g. multiple by steradians), for MAIA have dB-degrees^2**

**z\_tps <- tps$z**

**norm <- sum(z\_tps, na.rm=T) \* cell\_size**

**## your integrand**

**integrand\_tps <- z\_tps**

**## get numerical integral by summation:**

**volume\_tps <- sum(integrand\_tps, na.rm=T) \* cell\_size**

**volume\_tps = formatC(volume\_tps, digits = 2, format = "f")**

**#-----------------------------------------------------------**

**#===========================================================**

**############ 2d heatmap plot with MS value ##################**

**#============================================================**

**## uncheck below to obtain 2D heatmap of newly interpolated data**

**#print(plot\_tps)+labs(x = "x (degrees)", y = "y (degrees)")+**

**# ggtitle(paste("MAIA MS =",MAIAmeanthresh,"(dB)"))**

**#============================================================**

**###################### 3d plot ##############################**

**#============================================================**

**#factorise the different colours on the 3D plot**

**col\_tps <- palette[cut(tps$z, breaks = b)]**

**### use rgl package to generate 3D HOV**

**plot\_3D\_tpsvol <- rgl::persp3d(tps$x,tps$y,tps$z, color=col\_tps,**

 **xlim = c(xmin,xmax), ylim = c(ymin,ymax), zlim = c(0,40),**

 **xlab ="",ylab ="", zlab ="", axes=F, specular="gray60", ## specular changes colour of sunshine on plot**

 **sub="", main="", alpha = 1,**

 **aspect = c(100, 100, 40)) # changes axis aspect ratios**

**view3d(theta = 0, phi = -50) # change initial viewing angle of 3D plot**

**axes3d(c('x--','y--','z-+')) # change position of 3D axis (back or front)**

**title3d(xlab = "x (degrees)", line=1.5, cex=1.2)**

**title3d(ylab = "y (degrees)", line=3, cex=1.2)**

**mtext3d("[dB]", "z-+", line = 4, cex=1.2)**

**par3d(windowRect = c(0, 31, 769, 679)) # change scale of display window for 3D plot**

**#### print volume calculated onto plot title separately**

**bgplot3d({**

 **plot.new()**

**title(main = paste("Vol =",volume\_tps, "(dB-degrees^2)",**

 **"\n MAIA MS =",MAIAmeanthresh,"dB"),**

 **line=-6, cex.main=1.9)**

**})**

**##############################**

**### save a snapshot ##########**

**#rgl.snapshot("fig.png")**

**##############################**

**### save interactive HTML page**

**##############################**

**#browseURL(paste("file://", writeWebGL(dir=file.path("C:/file\_path\_here"), width=500), sep=""))**

**#################### End of program #############################**

**#############################################################**