FLIRJPG_Convert.R

GlennTattersall

Tue Jul 19 19:15:46 2016

```
############ TABLE OF CONTENTS ##############################
# 0.  Define all libraries and functions
# 1.  File Handling - manually enter file name as variable f
# 2.  Extract meta-tags from thermal file
# 3.  Set variables for use in temperature conversion from raw
# 4.  Extract binary raw data using exiftool and imagemagick
# 5. Export plain plot (png file) of the converted thermal image.
# 6. Export temperature data csv
# 7. Composite Plot - if width > height (i.e. landscape style)
# 8, Composite Plot - if width < length (i.e. portrait style)
############ END OF CONTENTS ################################

#0.  Define all libraries and functions
library(fields)
```

```
## Loading required package: spam
```

```
## Loading required package: grid
```

```
## Spam version 1.0-1 (2014-09-09) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: maps
```

```
##
## Attaching package: 'fields'
```

```
## The following object is masked from 'package:maps':
##
##     ozone
```

```
library(Thermimage)
library(png)
library(tools)
thermal.palette<-palette.choose("ironbow")
# can choose form "flir","ironbow"...

#1.  File Handling - create file loop or manually enter file name  ####
wd<-"~/Dropbox/R/MyProjects/FLIRJPGConvert"
wd<-"/Users/GlennTattersall/Dropbox/R/MyProjects/FLIRJPGConvert"

imgDir<-paste(wd, "/img", sep="")
outDir<-paste(wd, "/out", sep="")
txtDir<-paste(outDir, "/txt", sep="")
pngDir<-paste(outDir, "/png", sep="")
pdfDir<-paste(outDir, "/pdf", sep="")

dir.create(txtDir, showWarnings = F, recursive = FALSE, mode = "0777")
dir.create(pngDir, showWarnings = F, recursive = FALSE, mode = "0777")
dir.create(pdfDir, showWarnings = F, recursive = FALSE, mode = "0777")

setwd(imgDir)
getwd() #show working directory
```

```
## [1] "/Users/GlennTattersall/Dropbox/R/MyProjects/FLIRJPGConvert/img"
```

```
# Find any files with .jpg or .JPG ending
l.files<-list_files_with_exts(imgDir, full.names=FALSE, c("jpg","JPG","jpeg","JPEG"))

fn<-l.files[1]

for(fn in l.files[2]) # Delete [2] to force loop through all files
{

f<-paste(wd, "/img/", fn, sep="")
fname<-unlist(strsplit(fn,"[/]"))
fname<-fname[length(fname)]
f.root<-substr(fname,1,nchar(fname)-4)
# text output file name root, without .jpg
f.temperature<-paste(outDir, "/txt/", f.root,"_temperature.csv", sep="")
f.png<-paste(outDir, "/png/", f.root,".png", sep="")
f.composite<-paste(pdfDir, "/", f.root, "_composite.pdf", sep="")
finfo = file.info(fn)
finfo$size  # how many bytes in the file

graphics.off()
cat('Analysing file:', f)
cat('\n')


#2. Extract meta-tags from thermal vid file ####
cams<-flirsettings(f, exiftool="installed", camvals="")

#3. Set variables for calculation of temperature values from raw A/D sensor data  ####
```

```
op<-options(digits.secs=3)
w<-NULL;h<-NULL
Emissivity<-  cams$Info$Emissivity      # Image Saved Emissivity - should be ~0.95 or 0.96
ObjectEmissivity<-  0.96                # Object Emissivity - should be ~0.95 or 0.96
dateOriginal<-cams$Dates$DateTimeOriginal
print(dateOriginal)
dateModif<-   cams$Dates$FileModificationDateTime
dateCreate<-cams$Dates$CreateDate
dateOriginal[is.na(dateOriginal)]<-dateCreate
PlanckR1<-    cams$Info$PlanckR1           # Planck R1 constant for camera
PlanckB<-     cams$Info$PlanckB            # Planck B constant for camera
PlanckF<-     cams$Info$PlanckF            # Planck F constant for camera
PlanckO<-     cams$Info$PlanckO            # Planck O constant for camera
PlanckR2<-    cams$Info$PlanckR2           # Planck R2 constant for camera
OD<-          cams$Info$ObjectDistance        # object distance in metres
FD<-          cams$Info$FocusDistance         # focus distance in metres
ReflT<-       cams$Info$ReflectedApparentTemperature      # Reflected apparent temperature
AtmosT<-      cams$Info$AtmosphericTemperature      # Atmospheric temperature (for loss across distance
IRWinT<-      cams$Info$IRWindowTemperature   # IR Window Temperature  20
IRWinTran<-   cams$Info$IRWindowTransmission     # IR Window transparency 1
RH<-          cams$Info$RelativeHumidity      # Relative Humidity  50
h<-           cams$Info$RawThermalImageHeight # sensor height (i.e. image height)
w<-           cams$Info$RawThermalImageWidth  # sensory width (i.e. image width)
# Set thermal image frame parameters here w = width, h = height
# you must write these values in or the script will not work!
if(is.null(w)) w<-160
if(is.null(h)) h<-120
l<-w*h


#4. Extract binary data from image using readflirJPG ####
d<-readflirJPG(f, exiftool="installed")
d<-matrix(d, nrow=h)
d<-rotate270.matrix(d)
image.plot(d, useRaster=T, col=thermal.palette)

# 5. Convert binary data to temperature ####

# Consider whether you should change any of the following: ObjectEmissivity, OD, RH, ReflT,
# AtmosT, IRWinT, IRWinTran?
temperature<-raw2temp(d,ObjectEmissivity,OD,ReflT,AtmosT,IRWinT,IRWinTran,RH,
                      PlanckR1,PlanckB,PlanckF,PlanckO,PlanckR2)
colnames(temperature)<-NULL
rownames(temperature)<-NULL

t.sum<-summary(as.vector(temperature))
# Summary temperature data on the entire image
centre.point<-temperature[floor(w/2), floor(h/2)]
# Centre.point is equivalent to the centre spot ROI typical of FLIR images.
# Its value will be plotted below, but the symbol won't be placed on the graph
boxsize<-0.05
centre.box<-matrix(temperature[seq(w/2-boxsize*w, w/2+boxsize*w,1),
                               seq(h/2-boxsize*h,h/2+boxsize*h,1)])
```

```
centre.box<-temperature[-c(1:(w/2-boxsize*w), (w/2+boxsize*w):w),
                        -c(1:(h/2-boxsize*h), (h/2+boxsize*h):h)]

centre.box.m<-mean(centre.box)

# Define obj list for calling bquote in plots.
# will allow for different font format in plots within the same expression
obj<-list(fname=fn, timestamp=strftime(dateOriginal,
                                       origin="1970-01-01", format="%Y-%m-%d %H:%M:%OS", usetz=TRUE),
          min=round(t.sum[1], 1), max=round(t.sum[6], 1), mean=round(t.sum[4], 1),
          sd=round(sd(as.vector(temperature)),2), box=round(mean(centre.box), 1),
          spot=round(centre.point, 1))

# 6. Export PNG plot ####
png(f.png, width=8, height=8, units="in", res=300)
  image.plot(temperature, useRaster=TRUE, bty="n", col=thermal.palette,
           xlab="", ylab="", xaxt="n", yaxt="n",  asp=h/w, legend.shrink=0.72, legend.cex=0.85)
dev.off()

# 7. Export Temperature Data ####
write.csv(rotate90.matrix(temperature), f.temperature, row.names=FALSE)
# write to csv for opening up in ImageJ


# 8. Composite Plot - if width > height (i.e. landscape style) ####
# Optimised plot spacing if final composite image w:h ratio is 1.409
#graphics.off()
try(par(bg = "white"))
if(w>=h)
{
  pdf(file=f.composite, width=8, height=8/1.409, compress=F)
  split.screen(rbind(c(0.01,0.99,0.85,0.99), c(0.01,0.99,0.001,0.84)))
  try(par(mar=c(0,0,0,0)))
  screen(1)
  plot(0,1, type="n", ylab="", bty="n", xlim=c(0,1), ylim=c(0,1), xaxt="n", yaxt="n")
  rect(0,0,1,1, density=0, col="grey")
  text(0.5,0.75, bquote(bold("Filename: ") ~.(obj$fname)))
  text(0.5,0.2,  bquote(bold("Timestamp: ") ~.(obj$timestamp)), cex=0.9)
  split.screen(rbind(c(0.001,0.999,0.335,0.99), c(0.001,0.999,0.01,0.33)), screen=2)
  split.screen(rbind(c(0.15,0.70,0.01,0.99), c(0.701,0.99,0.01,0.99)), screen=3)
  screen(5)
  par(mar=c(0,0,0,0))
  image(temperature, add=FALSE, useRaster=TRUE, bty="n", col=thermal.palette,
        xlab="", ylab="", xaxt="n", yaxt="n",
        zlim=c(min(na.omit(temperature)), max(na.omit(temperature))),
        asp=h/w)
  rect(0,0,1,1)
  rect(0.5-boxsize,0.5-boxsize,0.5+boxsize,0.5+boxsize, border="grey")

  screen(6)
  par(mar=c(0,0,2,0))
  image.plot(legend.only=TRUE, col=thermal.palette, legend.cex=1,
             legend.shrink=1, smallplot=c(0.1,0.2,0,1),
```

```
            zlim=c(min(na.omit(temperature)), max(na.omit(temperature))))
  text(0.87,0.65, bquote(("°C")), xpd=TRUE, crt=90)
  split.screen(rbind(c(0.1,0.45,0.01,0.99), c(0.46,0.95,0.01,0.99)), screen=4)
  screen(7)
  par(mar=c(2,2,1,0))
  par(mgp=c(1,0.5,0))
  plot(density(temperature, adjust=0.1), main="", ylab=bquote(bold("No. Pixels")), xlab="",
       bty="n", yaxt="n", xaxt="n", xlim=c(t.sum[1], t.sum[6]),  col="black")
  axis(side=1, xaxp=c(floor(t.sum[1]), ceiling(t.sum[6]), 5), cex.axis=0.8, pos=0)
  polygon(density(temperature, adjust=0.1), col='black')
  rect(min(centre.box),-100, max(centre.box), 100, border="red", lwd=2)

  screen(8)
  par(mar=c(1,4,1,1))
  par(mgp=c(1,0,0))
  plot(0,1, type="n", ylab=bquote(bold("Statistics")), xlim=c(0,1),
       ylim=c(0,1), xaxt="n", yaxt="n")
  text(0,0.85,   cex=1.2, adj=c(0,NA), bquote(bold("Min:  ") ~.(obj$min)))
  text(0.6,0.85, cex=1.2, adj=c(0,NA), bquote(bold("Max:  ") ~.(obj$max)))
  text(0,0.55,   cex=1.2, adj=c(0,NA), bquote(bold("Mean: ") ~.(obj$mean)))
  text(0.6,0.55, cex=1.2, adj=c(0,NA), bquote(bold("SD:   ") ~.(obj$sd)))
  text(0,0.25,   cex=1.2, adj=c(0,NA), bquote(bold("Box:  ") ~.(obj$box)))
  text(0.6,0.25, cex=1.2, adj=c(0,NA), bquote(bold("Spot: ") ~.(obj$spot)))
  close.screen()
  close.screen(all.screens=TRUE)
  dev.off(dev.cur())
}



graphics.off()
#try(par(def.par))
try(par(bg = "white"))
# 9, Composite Plot - if width < length (i.e. portrait style) ####
# Optimised plot spacing if final composite image w:h ratio is 1.409
if(w<h)
{
  pdf(file=f.composite, width=8, height=8/1.409, compress=F)
  split.screen(rbind(c(0.01,0.99,0.85,0.99), c(0.01,0.99,0.1,0.83)))
  par(mar=c(0,0,0,0))
  screen(1)
  plot(0,1, type="n", ylab="", bty="n", xlim=c(0,1), ylim=c(0,1), xaxt="n", yaxt="n")
  rect(0,0,1,1, density=0, col="grey")
  text(0.5,0.75, bquote(bold("Filename: ") ~.(obj$fname)))
  text(0.5,0.2,  bquote(bold("Timestamp: ") ~.(obj$timestamp)), cex=0.9)
  split.screen(rbind(c(0.05,0.3,0.01,0.99), c(0.31,0.7,0.01,0.99), c(0.71,0.95,0.01,0.99)),
               screen=2)
  screen(4)
  par(mar=c(0,0,0,0))
  image(temperature, add=FALSE, useRaster=TRUE, bty="n", col=ironbowpal,
        xlab="", ylab="", xaxt="n", yaxt="n",
        zlim=c(min(na.omit(temperature)), max(na.omit(temperature))),
        asp=h/w)
```

```
  rect(0,0,1,1)
  screen(5)
  par(mar=c(0,0,2,0))
  image.plot(legend.only=TRUE, col=ironbowpal, legend.cex=1,
             legend.shrink=1, smallplot=c(0.1,0.3,0,1),
             zlim=c(min(na.omit(temperature)), max(na.omit(temperature))))
  text(0.9,0.5, bquote(bold("°C")), xpd=TRUE, crt=90)
  screen(3)
  split.screen(rbind(c(0.01,0.99,0.5,0.99), c(0.01,0.99,0.01,0.49)), screen=3)
  screen(6)
  par(mar=c(2,2,0,1))
  par(mgp=c(1,0.5,0))
  plot(density(temperature, adjust=0.1), main="", ylab=bquote(bold("No. Pixels")), xlab="",
       bty="n", yaxt="n", xaxt="n", xlim=c(t.sum[1], t.sum[6]),  col="black")
  axis(side=1, xaxp=c(floor(t.sum[1]), ceiling(t.sum[6]), 2), cex.axis=0.8, pos=0)
  polygon(density(temperature, adjust=0.1), col='black')
  screen(7)
  plot.new()
  par(mar=c(0,2,0,1))
  par(mgp=c(1,0.5,0))
  plot(0,1, type="n", ylab=bquote(bold("Summary Statistics")), xlim=c(0,1),
       ylim=c(0,1), xaxt="n", yaxt="n")
  text(0.05,0.86, cex=0.9, adj=c(0,NA), bquote(bold("Min:  ") ~.(obj$min)))
  text(0.05,0.72, cex=0.9, adj=c(0,NA), bquote(bold("Max:  ") ~.(obj$max)))
  text(0.05,0.58, cex=0.9, adj=c(0,NA), bquote(bold("Mean: ") ~.(obj$mean)))
  text(0.05,0.44, cex=0.9, adj=c(0,NA), bquote(bold("SD:   ") ~.(obj$sd)))
  text(0.05,0.30, cex=0.9, adj=c(0,NA), bquote(bold("Box:  ") ~.(obj$box)))
  text(0.05,0.16, cex=0.9, adj=c(0,NA), bquote(bold("Spot: ") ~.(obj$spot)))
  close.screen()
  close.screen(all.screens=TRUE)
  dev.off(dev.cur())
}




} # end of loop for f in l.files
```

```
## Analysing file: /Users/GlennTattersall/Dropbox/R/MyProjects/FLIRJPGConvert/img/IR_2412.jpg
## [1] "2013-05-09 22:22:23"
```

```
# Plot temperature image using ggplot2
library(ggplot2)
library(reshape2)
d<-reshape2::melt(temperature)

p<-ggplot2::ggplot(d, aes(Var1, Var2))+
  geom_raster(aes(fill=value))+coord_fixed()+
  scale_fill_gradientn(colours=ironbowpal)+
  theme_void()+
  theme(legend.key.height=unit(2, "cm"), legend.key.width=unit(0.5, "cm"))
library(grid)
```


35.0
32.5
30.0
27.5
25.0