Khyati Naik: Data 605 - HW4

Sep 23, 2023

```
library(imager)
```

```
## Loading required package: magrittr
```

```
##
## Attaching package: 'imager'
```

```
## The following object is masked from 'package:magrittr':
##
##     add
```

```
## The following objects are masked from 'package:stats':
##
##     convolve, spectrum
```

```
## The following object is masked from 'package:graphics':
##
##     frame
```

```
## The following object is masked from 'package:base':
##
##     save.image
```

```
library(jpeg)
library(EBImage)
```

```
##
## Attaching package: 'EBImage'
```

```
## The following objects are masked from 'package:imager':
##
##     channel, dilate, display, erode, resize, watershed
```

```
# Set the directory and file name prefix
directory <- "K:/khyati/cuny/605/hw4/jpg/"
file_prefix <- "RC_2500x1200_2014_us_"
file_extension <- ".jpg"

# Specify the number of images
num <- 17
```

```r
# Define a vector to store file paths
file_paths <- character(num)

# Generate the file paths
for (i in 1:num) {
  file_paths[i] <- paste0(directory, file_prefix, i, file_extension)
}
```

Above code reads the images from the folder.

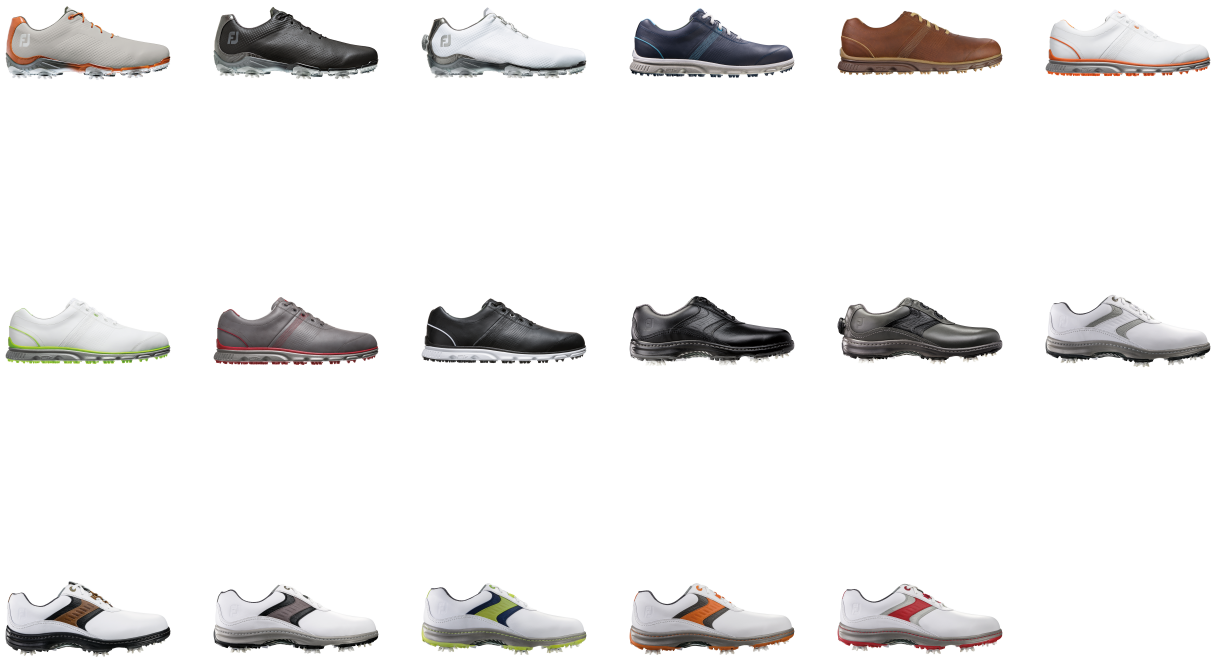
```r
# Image processing parameters
height <- 1200
width <- 2500
scale <- 20
```

```r
# Initialize an empty array to store images
im <- array(0, dim = c(num, height/scale, width/scale, 3))

# Read and preprocess the images
for (i in 1:num) {
  img <- readJPEG(file_paths[i])
  img_resized <- resize(img, height/scale, width/scale)
  im[i,,,] <- img_resized
}
```

```r
# Define a function to plot JPEG images
plot_jpeg <- function(path, add = FALSE) {
  jpg <- readJPEG(path, native = TRUE) # Read the file
  res <- dim(jpg)[2:1] # Get the resolution, [x, y]
  if (!add) # Initialize an empty plot area if add == FALSE
    plot(1, 1, xlim = c(1, res[1]), ylim = c(1, res[2]), asp = 1, type = 'n',
         xaxs = 'i', yaxs = 'i', xaxt = 'n', yaxt = 'n', xlab = '', ylab = '', bty = 'n')
  rasterImage(jpg, 1, 1, res[1], res[2])
}
```

Above code plots the images.

```r
# Set up plot parameters
par(mfrow = c(3, 6))
par(mai = c(.01, .01, .01, .01))

# Plot the first images
for (i in 1:num) {
  plot_jpeg(file_paths[i])
}
```



```r
# Initialize an empty matrix 'flat' to store flattened pixel values
flat <- matrix(0, num, prod(dim(im)))

# Flatten the pixel values and store in 'flat'
for (i in 1:num) {
  r <- as.vector(im[i,,,1])
  g <- as.vector(im[i,,,2])
  b <- as.vector(im[i,,,3])
  flat[i,] <- t(c(r, g, b))
}
```

```r
# Convert 'flat' matrix to a data frame 'shoes'
shoes <- as.data.frame(t(flat))
```

```r
scaled=scale(shoes, center = TRUE, scale = TRUE)
Sigma_=cor(scaled)
myeigen=eigen(Sigma_)
```

```r
cum_var <- cumsum(myeigen$values) / sum(myeigen$values)
cum_var
```

```
##  [1] 0.6928202 0.7940449 0.8451073 0.8723847 0.8913841 0.9076338 0.9216282
##  [8] 0.9336889 0.9433872 0.9524455 0.9609037 0.9688907 0.9765235 0.9832209
## [15] 0.9894033 0.9953587 1.0000000
```

```r
thresh_var <- min(which(cum_var >= .80))
thresh_var
```

```
## [1] 3
```

Above outputs shows that 80% variability can be seen at position 2. Hence, we find eigen shoes at [,2].

```r
# Rescale the eigenimages
scaling <- diag(myeigen$values[1:thresh_var]^(-1/2)) / sqrt(nrow(scaled) - 1)
eigenshoes <- scaled %*% myeigen$vectors[, 1:thresh_var] %*% scaling

# Reshape the eigenimages
eigen_images <- array(eigenshoes[, 2], dim = c(60, 125, thresh_var))

# Save the second eigenimage as a PNG file
png("eigenimage.png")
image(eigen_images[, , 2], colormode = "RGB")
dev.off()
```

```
## pdf
##   2
```

```r
# Display the saved image
img <- load.image("eigenimage.png")
plot(img)
```