```
# Load image

setwd("c:/repos/patternrecognition/project/ImageRecognition" )
path <- "ImageRecognition/images/ingalls-rink-1.png"
image <- readImage(path)

# Display properties

print(image)

# Output:
#
# Image
#   colorMode    : Color
#   storage.mode : double
#   dim          : 256 256 4
#   frames.total : 4
#   frames.render: 1
#
# imageData(object)[1:5,1:6,1]
#           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
# [1,] 0.2039216 0.2039216 0.2000000 0.2039216 0.2039216 0.2000000
# [2,] 0.2039216 0.2039216 0.2039216 0.2078431 0.2078431 0.2078431
# [3,] 0.2039216 0.2078431 0.2078431 0.2078431 0.2117647 0.2156863
# [4,] 0.2078431 0.2078431 0.2039216 0.2078431 0.2078431 0.2196078
# [5,] 0.2039216 0.2078431 0.2078431 0.2078431 0.2078431 0.2196078

# Convert image to Grayscale

colorMode(image) <- Grayscale

print(image)

# Image
#   colorMode    : Grayscale
#   storage.mode : double
#   dim          : 256 256 4
#   frames.total : 4
#   frames.render: 4
#
# imageData(object)[1:5,1:6,1]
#           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
# [1,] 0.2039216 0.2039216 0.2000000 0.2039216 0.2039216 0.2000000
# [2,] 0.2039216 0.2039216 0.2039216 0.2078431 0.2078431 0.2078431
# [3,] 0.2039216 0.2078431 0.2078431 0.2078431 0.2117647 0.2156863
# [4,] 0.2078431 0.2078431 0.2039216 0.2078431 0.2078431 0.2196078
# [5,] 0.2039216 0.2078431 0.2078431 0.2078431 0.2078431 0.2196078

# Apply edge detection
```

```
laplacianFilter <- matrix(1, nrow = 3, ncol = 3)
laplacianFilter[2, 2] <- -8
filteredImage <- filter2(image, laplacianFilter)

# Convert image to matrix

channelCount <- 4
imageMatrix <- matrix(image, ncol = channelCount, byrow = TRUE)
str(imageMatrix)

# num [1:65536, 1:4] 0.204 0.204 0.204 0.2 0.184 ...

# Run PCA

pc <- princomp(imageMatrix)

# List of 7
#  $ sdev    : Named num [1:4] 0.6228 0.0859 0.0449 0.0284
#   ..- attr(*, "names")= chr [1:4] "Comp.1" "Comp.2" "Comp.3" "Comp.4"
#  $ loadings: loadings [1:4, 1:4] -0.492 -0.502 -0.506 -0.499 0.702 ...
#   ..- attr(*, "dimnames")=List of 2
#   .. ..$ : NULL
#   .. ..$ : chr [1:4] "Comp.1" "Comp.2" "Comp.3" "Comp.4"
#  $ center  : num [1:4] 0.646 0.645 0.642 0.643
#  $ scale   : num [1:4] 1 1 1 1
#  $ n.obs   : int 65536
#  $ scores  : num [1:65536, 1:4] 0.878 0.876 0.884 0.891 0.909 ...
#   ..- attr(*, "dimnames")=List of 2
#   .. ..$ : NULL
#   .. ..$ : chr [1:4] "Comp.1" "Comp.2" "Comp.3" "Comp.4"
#  $ call    : language princomp(x = imageMatrix)
#  - attr(*, "class")= chr "princomp"

# Plot PC loadings

library(lattice)
library(reshape2)

pc.load <- cbind(pc$loadings[, 1:channelCount])
colnames(pc.load) <- c("PC1", "PC2", "PC3", "PC4")
pc.df <- melt(pc.load)
xyplot(value ~ Var1, data = pc.df, group = Var2, type = "l",
       ylab = "PC Loadings", xlab = "Spectral Bands",
       auto.key = list(corner = c(0.98, 0.98), points = FALSE, lines = TRUE),
       panel = function(x, y, ...) {
    panel.grid(h = -1, v = -1)
    panel.xyplot(x, y, ...)
    panel.abline(h = 0, lty = "dashed")
```

```
})

# Plot variability for each principal component

PC1 <- (pc$sdev[1] ^ 2) * (pc$loadings[, 1] ^ 2) / diag(var(imageMatrix))
PC2 <- (pc$sdev[2] ^ 2) * (pc$loadings[, 2] ^ 2) / diag(var(imageMatrix))
PC3 <- (pc$sdev[3] ^ 2) * (pc$loadings[, 3] ^ 2) / diag(var(imageMatrix))
PC4 <- (pc$sdev[4] ^ 2) * (pc$loadings[, 4] ^ 2) / diag(var(imageMatrix))
PCSums <- PC1 + PC2 + PC3 + PC4
pcVar <- melt(cbind(PC1, PC2, PC3, PC4, "PC Sums" = PCSums))
xyplot(value ~ Var1, data = pcVar, group = Var2, type = "l",
       ylab = "Portion of Explained Variability", xlab = "Spectral Bands",
       auto.key = list(corner = c(0.45, 0.5), points = FALSE, lines = TRUE),
       panel = function(x, y, ...) {
    panel.grid(h = -1, v = -1)
    panel.xyplot(x, y, ...)
    panel.abline(h = 1, lty = "dashed")
})

# Calculate percentage of variability for each principal component

round((as.numeric(pc$sdev) ^ 2) / sum(as.numeric(pc$sdev) ^ 2) * 100, 3)

# [1] 97.435  1.854  0.507  0.203

# Scree plot

xyplot((pc$sdev ^ 2) ~ 1:channelCount, pch = 20, cex = 3, alpha = 0.75, type = "b",
       xlab = "Spectral Bands", ylab = "Eigenvalues",
       panel = function(x, y, ...) {
    panel.grid(h = -1, v = -1)
    panel.xyplot(x, y, ...)
})

# Simple fair share rule

which((pc$sdev ^ 2) > (sum(pc$sdev ^ 2)) / length(pc$sdev))

# Comp.1
#      1

# Display PC1

img <- array(0, c(256, 256, channelCount))
for (i in 1:256) {
    for (j in 1:256) {
        n = (i - 1) * 256 + j
        img[i, j,] <- as.vector(pc$scores[n,])
    }
```

```
}

img1 <- array(0, c(256, 256, channelCount))
for (i in 1:channelCount)
    img1[,, i] <- img[,, i]

display(img1[,, 1], all = T, meth = 'r')
```