Diego Garcia-Olano
DAKD 2013
Assignment 3

We first choose the balloons image for the assignment and load the image into R via the following command.

```
>blns <- read.pnm("../DAKD/belanche/Lecture2-Labo/balloons.pgm")
>gray.nmatrix <- blns@grey;
```

We analyze it to get its image dimensions as follows:

```
> blns
Pixmap image
Type         : pixmapGrey
Size         : 480x640
Resolution   : 1x1
Bounding box : 0 0 640 480
```

We then create a new sampling grid (j.nx and j.ny) based off the prior code, deciding to follow a similar pattern as was used with the dog image. The dog image was 256 rows x 256 columns and its projection was made by dividing the image into a 32 x 32 grid with each segment of the grid accounting for a 8 x 8 portion of the original image. Accordingly, with the balloon image that is 480 rows by 640 columns, we decide to break it down into a 32 x 32 grid in which each segment of it would represent a 15 x 20 block which would be used for the projections of the image.

```
Nx.seg      <- 20;
Ny.seg      <- 15;
j.nx   <-  seq(Nx.seg,640, by=Nx.seg);
j.ny   <-  seq(Ny.seg,480, by=Ny.seg);
N      <- length(j.x);
```

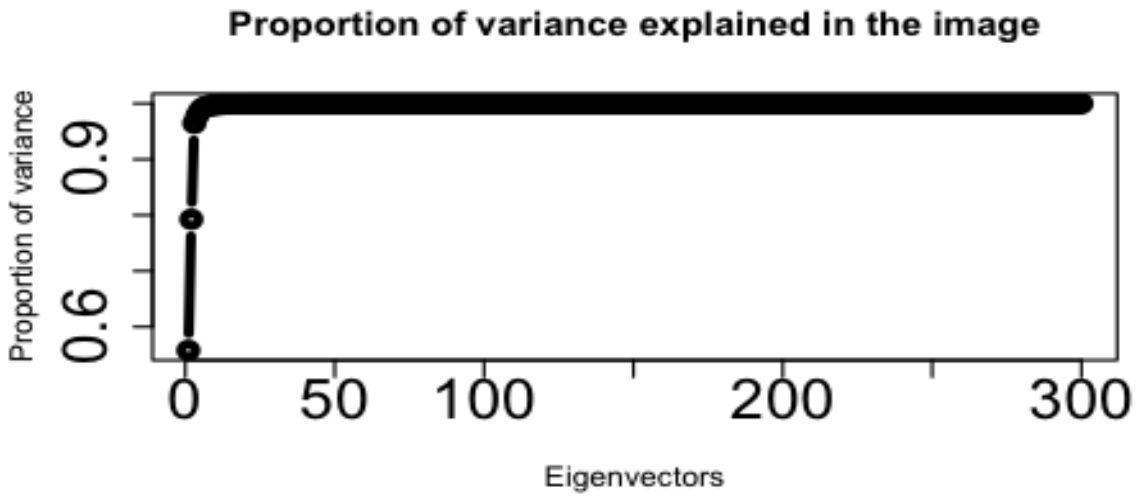
We then need to create the image segments from the gray balloon image.
In order to do so we change the function create.image.segements as follows:

```
create.image.rectanglesegments <- function(gray.nmatrix) {
#20 columns by 15 rows per square  ,
counter  <- 1;
seg.mean.nmatrix <- matrix(rep(0,Nx.seg*Ny.seg),  nrow=Ny.seg, ncol=Nx.seg);
seg.nmatrix      <- matrix(nrow=(Nx.seg*Ny.seg),ncol=(N*N));

      for (x in 1:N) {
        for (y in 1:N) {
              X <- gray.nmatrix[ (j.ny[x]-14):j.ny[x], (j.nx[y]-19):j.nx[y] ];
              seg.nmatrix[1:(Nx.seg*Ny.seg),counter] <- X;
```

```
            seg.mean.nmatrix <- X + seg.mean.nmatrix;
            counter <- counter + 1;
        }
    }

seg.mean.nmatrix <- 1/(N*N) * seg.mean.nmatrix;
C <- cov(t(seg.nmatrix));
return (list(seg.mean.matrix = seg.mean.nmatrix,  C = C, seg.matrix = seg.nmatrix));
}
```

The principle changes from the original square implementation are in defining the dimensions of seg.mean.matrix, seg.matrix, and the inner loop calculation for the matrix X. Seg.mean.matrix is now of dimension 15 x 20 to reflect the size of each segment of the grid, and seg.matrix is 300 x 1024, so that each column (all 32x32 of them) may store the pixels for each different 15x20 grid (ie, 300 values for each 1024 combinations). The calculation of X, the loop variable which stores the results for the current segment of the 32x32 grid being calculated, is made by grabbing the current 15x20 block from the original image, where x holds the current row and y holds the current column being processed. This point x,y is the location of the projection on the 32x32 result.

Seg.matrix is then transposed so that the prior 300 rows, which are the data for each 15x20 block, are now the columns (ie features) that we can then use to compute the variance, which is done with **cov(t(seg.nmatrix))**. We then plot this covariance matrix with the following function call: **plot.prop.variance(nResult$C);**



The function **plot.prop.variance** computes the principal component analysis of the segments matrix we provide as input and then shows how much each component's variance contributes to the cumulative variance of the overall image. In this case it's clear that with less than 10 eigenvectors, 6 to be exact, we have explained 99 % of the

variance. This finding would allow us to then select these principal components to use for feature selection if dimensionality reduction for visualization purposes was our goal, as is illustrated in this exercise. In fact, we could use the first three components alone as they account for 96 percent of the variance depending on how much information loss we deemed acceptable.

We then plot the original image as such
>plot(pixmapGrey(gray.nmatrix),main="Original image");

Original image

Finally, we then reconstruct the image with different proportions of the components/eigenvectors obtained from PCA to show visually how much of the image is accounted for using these different proportions of components (25,50,75, and 100% respectively). To do so we need to first slightly change the function, **image.segments.reconstruction**, to account for the rectangular dimensions of our image as follows:

```
image.segments.reconstruction
      <- function(seg.matrix, C, M.pcvec, seg.mean.matrix) {
      ….
      reconstr.seg.matrix[1:(Ny.seg*Nx.seg),n] <- x.hat;
      }
return (reconstr.seg.matrix);

}
```

We also similarly need to edit the function **plot.seg.matrix:**

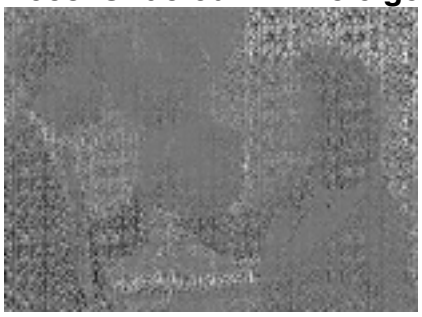
```
plot.seg.matrix <- function(seg.matrix,text) {
        gray.matrix <- matrix(ncol=640,nrow=480);
        counter <- 1;
          for (x in 1:N) {
            for (y in 1:N) {
              X <- matrix(seg.matrix[,counter],byrow=FALSE,ncol=Nx.seg,nrow=Ny.seg);
              gray.nmatrix[ (j.ny[x]-14):j.ny[x], (j.nx[y]-19):j.nx[y] ] <- X;
              counter <- counter + 1;
            }
          }
          plot(pixmapGrey(gray.nmatrix),main=text);
}
```

Finally to plot the different amounts of eigenvector use, and in turn showing the amount of information gained by the use of more components, we loop over those proportions, reconstructing the image with the current proportion of components and plotting them with our newly updated functions.

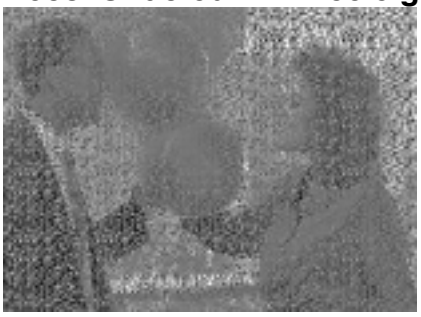
```
for (i in 1:length(M.pcvec)) {
  Reconstructed <- image.segments.reconstruction(nResult$seg.matrix,  nResult$C,
                          (Ny.seg*Nx.seg)*M.pcvec[i] ,  nResult$seg.mean.matrix);
  text <- paste("Reconstructed with ",(Ny.seg*Nx.seg)*M.pcvec[i]," eigenvectors",sep="");
  plot.seg.matrix(Reconstructed,text);
}
```

The final plots for 25, 50, 75 and 100 % use of the components obtained is as follows:

Reconstructed with 75 eigenvectors



Reconstructed with 150 eigenvectors



Reconstructed with 225 eigenvectors

Reconstructed with 300 eigenvectors