Eric Yang

304263623

Homework 3 + 4

1.

a.

```
function singularValues = eigenfaces(files, k, s)

%Convert between images and vectors
row = 64;
col = 64;
image_vector = @(Bitmap) double(reshape(Bitmap,1,row*col));
vector_image = @(Vec) reshape( uint8( min(max(Vec,0),255) ), row, col);
vector_render = @(Vec) imshow(vector_image(Vec));


for i = 1:s
    image = char(files(i));
    matrix = imread(image);
    faces(i,:) = image_vector(matrix);
end

vector_render(mean(faces)');
means = ones(n,1) * mean(faces);
X = (faces-means);

[U, S, V] = svds(cov(X), k);
singularValues = diag(S); %output the first k singular values
Eigenfaces = V;

matrix_image = @(A) uint8(round ( (A-min(A))/(max(A)-min(A))*255 ));
figure

x=ceil(sqrt(k)); %determine measurements of plot based on number of eigenfaces
for i=1:k
    subplot(x,x,i)
    vector_render(matrix_image(Eigenfaces(:,i)));
end
```

b.

```
5.9576e+005
3.5977e+005
1.9937e+005
1.8156e+005
1.4907e+005
1.2380e+005
8.1178e+004
7.7082e+004
6.1404e+004
5.8490e+004
4.8555e+004
4.4170e+004
4.1063e+004
3.8751e+004
3.4346e+004
3.2908e+004
2.9112e+004
2.8268e+004
2.5141e+004
2.3658e+004
2.2247e+004
2.0692e+004
1.9409e+004
1.8361e+004
1.7990e+004
1.6267e+004
1.6020e+004
1.5108e+004
1.3839e+004
1.3587e+004
```

c.

Average face for good:

Average face for evil:

d. good – evil:

e. 102 and 103 are a pair

f.

```
function result = goodOrEvil(image)

evil =['face014.bmp'  ,'face017.bmp'  ,'face018.bmp'  ,'face019.bmp'  ,'face026.bmp'  .........];

good = ['face000.bmp'   ,'face001.bmp'   ,'face002.bmp'   ,'face003.bmp'   ,'face004.bmp'......];

row = 64;
col = 64;
image_vector = @(Bitmap) double(reshape(Bitmap,1,row*col));
vector_image = @(Vec) reshape( uint8( min(max(Vec,0),255) ), row, col);
vector_render = @(Vec) imshow(vector_image(Vec));

for i = 1:122
    image = char(good(i));
    matrix = imread(image);
    faces(i,:) = image_vector(matrix);
end
vector_render(mean(faces)');
means = ones(n,1) * mean(faces);
X = (faces-means);

[U, S, V] = svds(cov(X), 30);
f0 = V;

for i = 1:47
    image = char(evil(i));
    matrix = imread(image);
    faces(i,:) = image_vector(matrix);
end
vector_render(mean(faces)');
means = ones(n,1) * mean(faces);
X = (faces-means);

[U, S, V] = svds(cov(X), 30);
f1 = V;

row = 64;
col = 64;
image_vector = @(Bitmap) double(reshape(Bitmap,1,row*col));
vector_image = @(Vec) reshape( uint8( min(max(Vec,0),255) ), row, col);
vector_render = @(Vec) imshow(vector_image(Vec));

matrix = imread(image);
f=double(matrix);
good = (f-f0);
evil = (f-f1);
result = (norm(evil) <= norm(good));
```

The function returns 01111011111101110111110110111011111011110111101. The function doesn't seem to work too well probably because a lot of the faces look similar. The evil and good average face are not that far apart.

2.

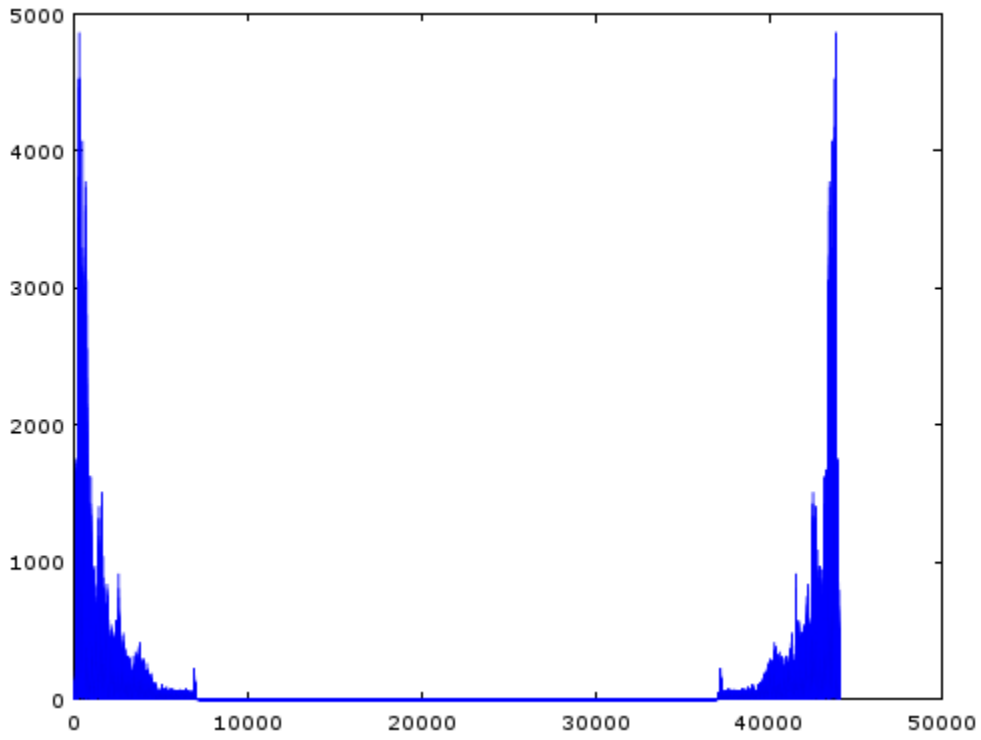
a. Fs is 44100hz sampling frequency which is the usual CD-quality sample rate.

b. y is 1411200 entries long.

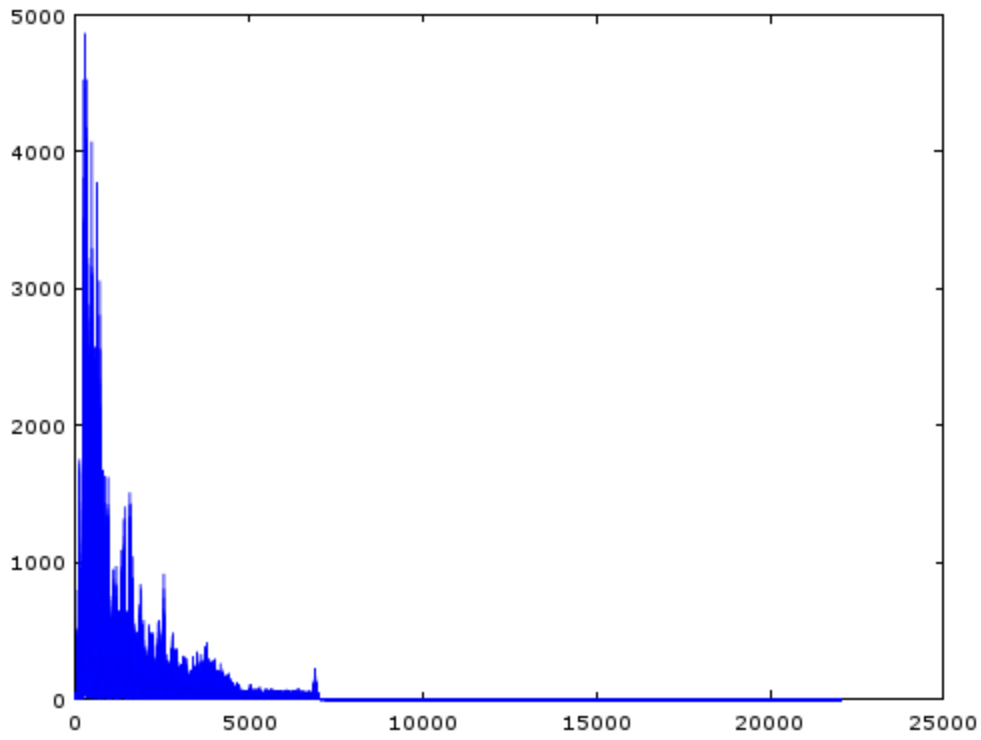
```
>> n = factor(1411200)
n =

   2   2   2   2   2   2   2   3   3   5   5   7   7
```

c. fft(y) takes 0.0419381 seconds when tested with tic; and toc; and it takes 0.040 seconds tested with profile

d.



e. Nyquist frequency = 22050



f.

```
function [spike_freq,spike_power] = top_spike(frequency_values,power_values)

spike = find(power_values == max(power_values));
while (frequency_values(spike) < 100)
  power_values(spike) = 0;
  spike = find(power_values == max(power_values));
endwhile
spike_freq = frequency_values(spike);
spike_power = power_values(spike);

```

Using the function top_spike, I found the top 4 spikes in the interval from 0 to 22050 (excluding frequencies below 100) to be at 265.39hz, 312.99hz, 234.21hz, 265.77hz.

g. The key is C# at frequency 265.39hz and there is no harmony as the next spike is at 312.99hz but 265.39(5/4) = 331.74hz.

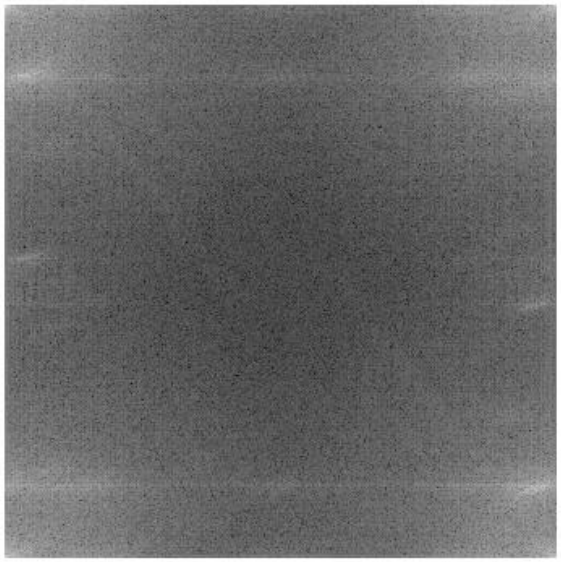
h. The top spike for untune is at 104.09hz and there is no harmony as the next spike is around 184.35hz and that is not a multiple of 5/4(104.09hz).

3.

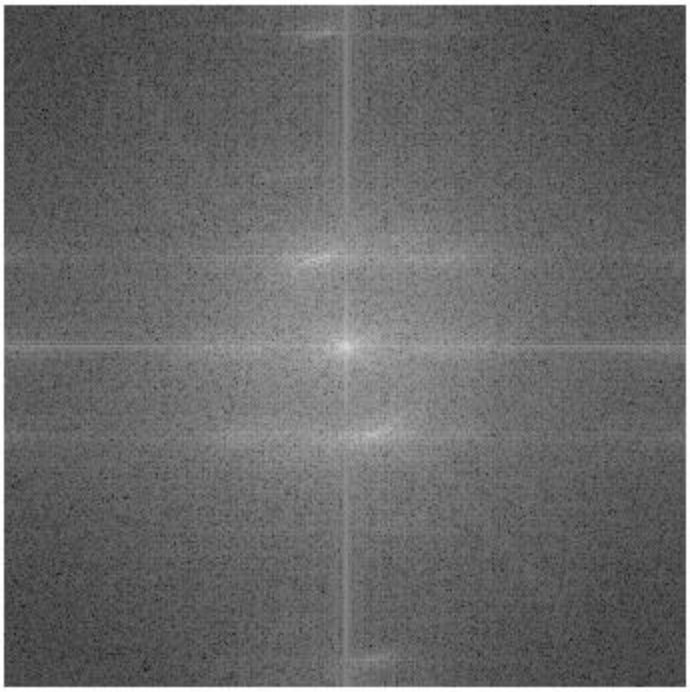
a.

```
>> image = imread('taeyeon.bmp')
image =

 Columns 1 through 14:

  178   189   205   222   222   210   187   152   151   176   215   232   230   195
  149   160   169   178   179   171   157   140   141   159   184   195   193   172
  146   142   143   146   146   142   137   134   136   141   153   159   158   155
  158   138   128   129   131   132   129   132   132   126   127   127   131   145
  158   154   161   176   185   179   162   153   144   136   145   153   159   165
  178   178   190   215   224   215   190   161   148   151   171   182   186   187
  205   200   213   242   249   236   207   162   150   166   195   209   209   206
  204   199   207   228   230   217   189   149   142   165   193   204   202   204
  198   191   192   200   194   183   163   140   139   152   174   183   182   179
  191   176   167   166   155   143   135   134   134   138   146   154   154   143
  170   152   139   136   136   133   128   128   129   129   140   142   137   132
  168   151   142   139   148   152   152   148   151   158   158   155   151   144
  178   165   162   163   179   186   188   181   186   196   186   182   180   172
  203   195   188   183   197   205   205   190   186   194   201   201   197   195
  188   192   192   192   199   203   202   184   177   180   194   200   197   193
  157   170   181   187   187   186   186   169   163   164   180   186   187   180
  141   143   143   142   150   152   151   139   143   159   178   185   182   170
```

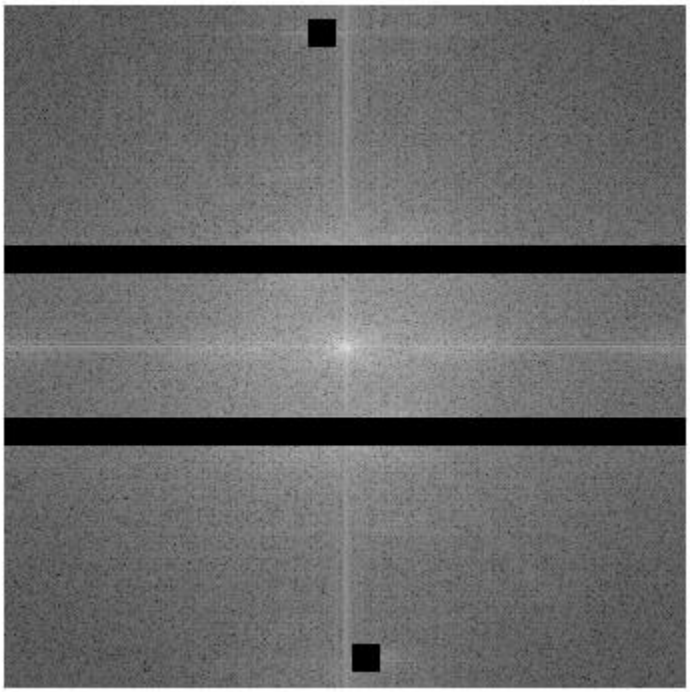
b.



c.



d.



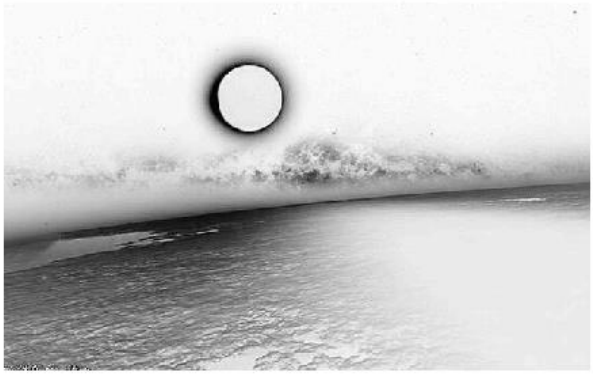
e. There seems to be some improvement when put together with the image before noise removal:

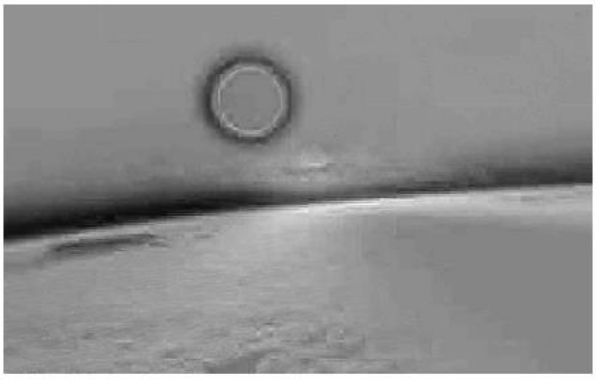
4.

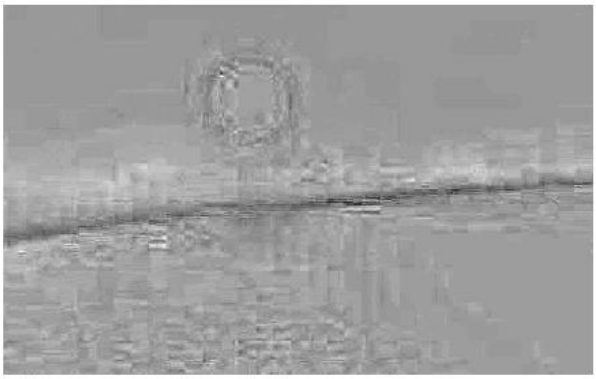
a.

```
function newPhoto = isFake(photo,pc)

A=imread(photo);
sizeA=size(A);
m=sizeA(1);
n=sizeA(2);

R=reshape(A,m*n,3);
CovR=cov(double(R));
[U,S,V]=svd(CovR);

PC=U(:,1:3);
ThirdPC = PC(:,3);
SecondPC=PC(:,2);
FirstPC = PC(:,1);
if pc == 1
   newPhoto=double(R)*double(FirstPC);
   newPhoto=reshape(newPhoto,m,n);
   imshow(newPhoto, []);
   end
if pc ==2
   newPhoto=double(R)*double(SecondPC);
   newPhoto=reshape(newPhoto,m,n);
   imshow(newPhoto, []);
end
if pc == 3
   newPhoto=double(R)*double(ThirdPC);
   newPhoto=reshape(newPhoto,m,n);
   imshow(newPhoto, []);
end
```

Ran my function isFake() on the photo and it seems to be fake due to many bright and dark spots in first photo:

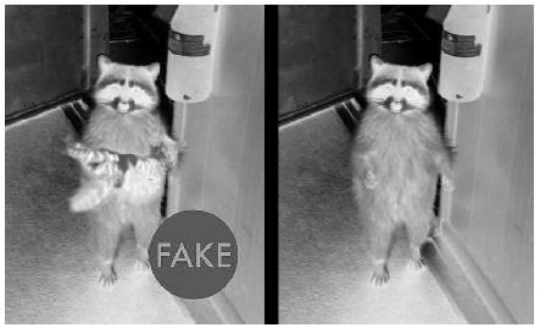


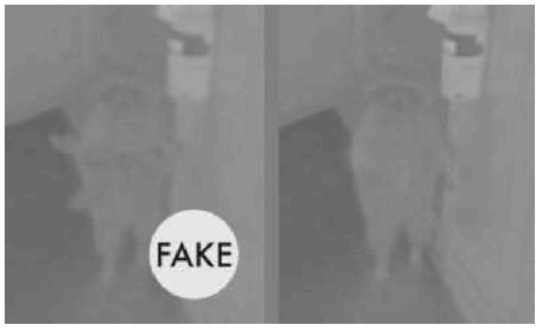


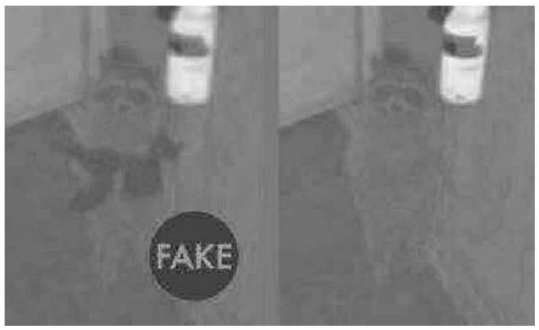


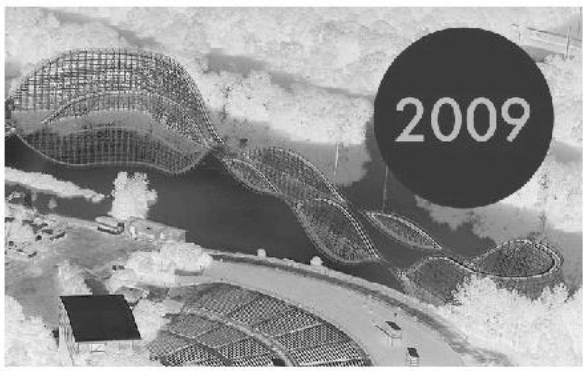
b. this shark attack photo also seems to be fake from the bright spots in the first photo:

c. In these photos, it seems the 1st and 3rd PC are better indicators of fake photos. The raccoon one doesn't show much in the 2nd PC but the 3rd PC shades the cat in black and the 1st shows inconsistent coloring of the cat in contrast to the raccoon so this photo is fake. The flooding one seems to be real. The 2nd and 3rd PC don't indicate anything is strange and the 1st shows a lot of bright white and black that seems to be just the color contrast between water and grass/trees.


FAKE


FAKE


FAKE


2009


2009