Assignment 2 - STAT 7730

Problem 2 - PCA for image patches

For this homework we will be working with 12×12 image patches. There are 100 grayscale images of outdoor scenes.

In [8]:
```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import os
import random as rnd
import math
```

We will first read all the images into numpy arrays

In [9]:
```
A = np.zeros((100,600,600))
X = np.zeros((144,100000))

pix_img = 589**2;

path = os.getcwd() + '\HW2\img\\'

for i in range(1,101):
    filname = 'img' + str(i)
    A[i-1,:,:] =  cv2.imread(path + filname, 0)
```

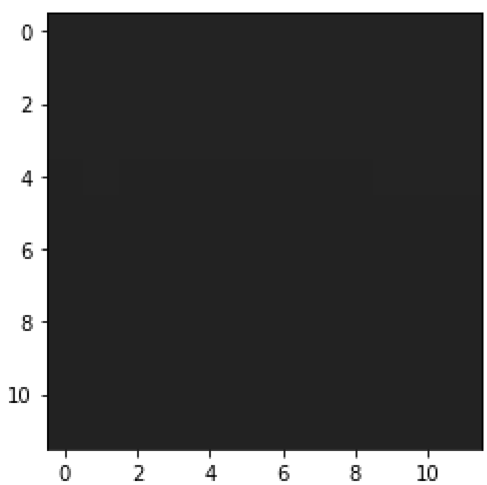
To estimate the covariance matrices of the 12×12 patches we will randomly choose $n = 10^5$ image patches from the database. Then we will compute all 144 eigenvectors of the covariance matrix sorted such that their corresponding eigenvalues are decreasing.

In [10]:
```
sampls = rnd.sample(range(0,100*(600-11)**2),100000)

for j in range(0,100000):
    i = sampls[j]
    num_img = math.floor(i/pix_img)
    n_pix = i % pix_img
    i_1 = math.floor(n_pix / 589)
    i_2 = n_pix % 589
    V = A[num_img,i_1:(i_1 + 12),i_2 : (i_2 + 12)]
    X[:,j] = V.flatten('F')


C = np.cov(X)
L, U = np.linalg.eig(C)
```

Some basic statistics about the values in the data:

In [11]:
```
print("Maximum and minimum values of pixel intensities",end = '')
print(" of the 100 images: \n\t Max: %d" % X.max())
print("\t Min: %d\n" % X.min())

print("Mean of all the pixels in the 100 images: %f\n" % np.ndarray.mean(X))
mu_vec = X.mean(axis = 1)
mu_vec = mu_vec.reshape((144,1))
mu_vec_1 = mu_vec.reshape((12,12)).T

plt.imshow(mu_vec_1.astype(int), cmap='gray', vmin=0, vmax=255)
print("What does the mean image patch look like?")
plt.show()

print("It looks like a dark square, almost black\n")
```

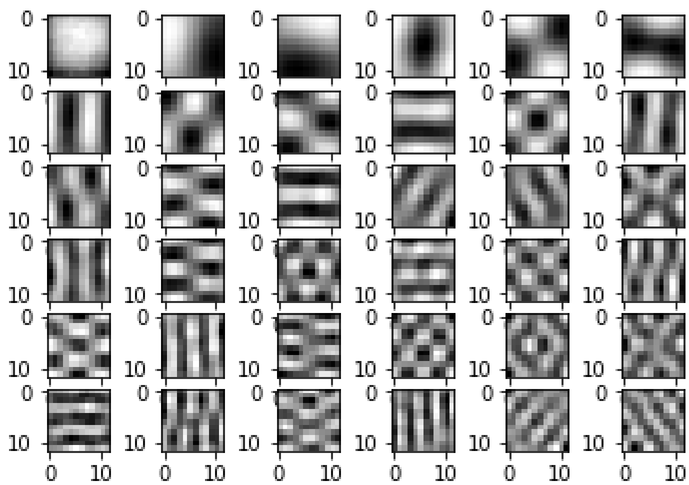
```
Maximum and minimum values of pixel intensities of the 100 images: 
	 Max: 255
	 Min: 2

Mean of all the pixels in the 100 images: 34.885862

What does the mean image patch look like?
```




```
It looks like a dark square, almost black
```

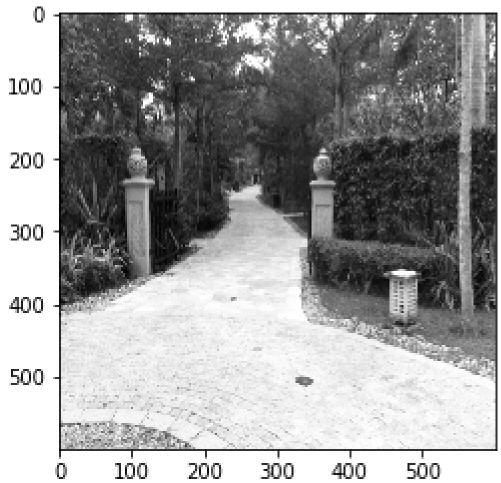
Now we will create a figure with the first 36 eigenvectors (in order, from left-to-right, then top-to-bottom) displayed as 12×12 image patches. We can observe the dominant modes of variation in the top left patch.

In [12]:
```
eig_vec = np.zeros((36,12,12))
img_eig = np.zeros((12*6,12*6))


fig, axs = plt.subplots(6, 6)

j = 0
for k in range(0,36):
    i = k % 36
    patch = U[:,k]
    patch = patch - patch.min()
    patch = patch*255/patch.max()
    patch = patch.reshape((12,12)).T
    patch = patch.astype(int)
    i_1 = math.floor(i/6)
    i_2 = i % 6
    axs[math.floor(k/6),k % 6].imshow(patch, cmap='gray', vmin=0, vmax=255)

plt.show()
```



In [13]:
```
sample_img = A[0,:,:]
sample_img = sample_img.astype(int)

print("We will use the first image of the database to work with")

plt.imshow(sample_img, cmap='gray')
plt.show()

sample_patches = np.zeros((144,2500));

for i in range(0,50):
    for j in range(0,50):
        patch = sample_img[(i*12):(i*12+12),(j*12):(j*12+12)]
        patch = patch.flatten('F');
        sample_patches[:,i*50+j] = patch;
```

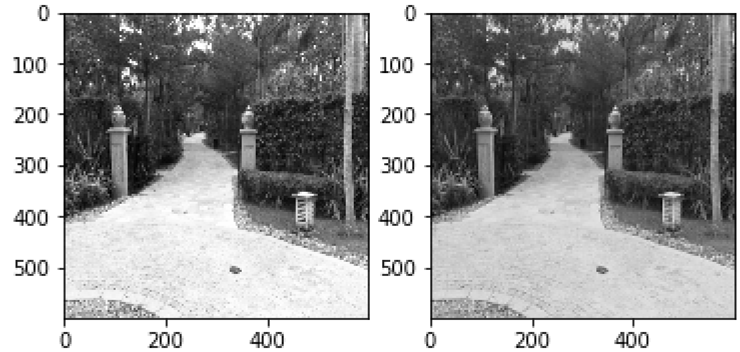
```
We will use the first image of the database to work with
```



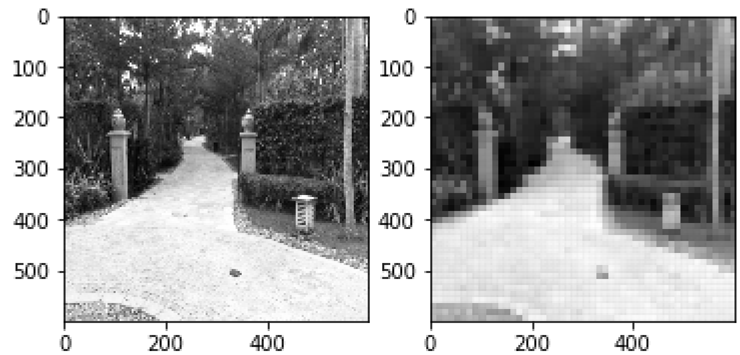
We finally show the PCA versions of the image. Observe that when $d \geq 10$, the PCA version is indeed very similar to the original one, and most of the quality is preserved.

In [14]:
```
for d in [20,1,3,5,10,144]:
    fig, axs = plt.subplots(1, 2)
    U_d = U[:,0:d]
    U_n = U_d.dot(U_d.T)
    X_0 = sample_patches - mu_vec.dot(np.ones((1,2500)))
    Z = U_n.dot(X_0) + mu_vec.dot(np.ones((1,2500)))
    Z_img = np.zeros((600,600))

    for i in range(0,50):
        for j in range(0,50):
            patch = Z[:,i*50+j];
            patch = patch.reshape((12,12)).T
            Z_img[(i*12):(i*12+12),(j*12):(j*12+12)] = patch

    Z_img = Z_img.astype(int);
    print("d = %d\tprincipal components" % d)
    axs[1].imshow(Z_img, cmap='gray')
    axs[0].imshow(sample_img, cmap='gray')
    plt.show()
```

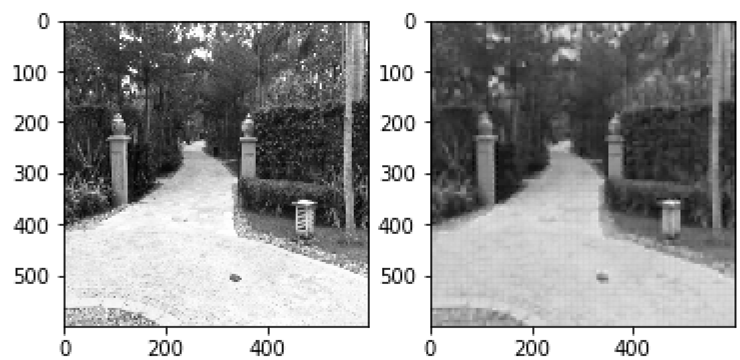
```
d = 20  principal components
```



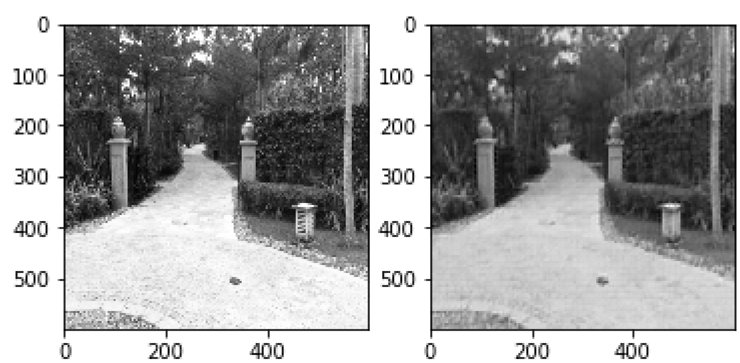

```
d = 1   principal components
```



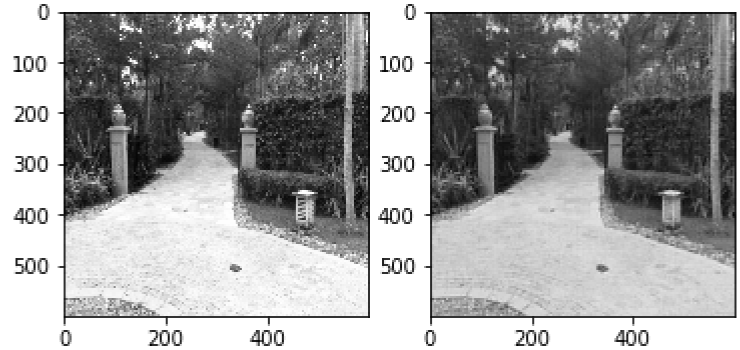

```
d = 3   principal components
```




```
d = 5   principal components
```




```
d = 10  principal components
```



d = 144 principal components