EECS16A: Homework 4

Problem 5: Image Compression

In [1]:
```
%pylab inline
```

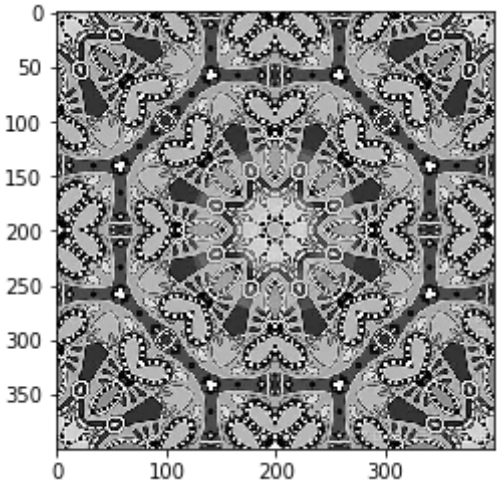
```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
import numpy as np
from scipy import ndimage as nd
from scipy import misc
from scipy import io
```

Part a

In [3]:
```
#Load Pattern Image
pattern = np.load('pattern.npy')
plt.imshow(pattern, cmap='gray', interpolation='nearest')
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x10d876ad0>
```



Use the command shape (http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html) to find the dimensions of the image. How many eigenvalues do you expect?

Run the code below to find the eigenvector and eigenvalues of `pattern` and sort them in descending order (first eigenvalue/vector corresponds to the largest eigenvalue)

In [4]:
```
eig_vals, eig_vectors = np.linalg.eig(pattern)
idx = (abs(eig_vals).argsort())
idx = idx[::-1]
eig_vals = eig_vals[idx]
eig_vectors = eig_vectors[:,idx]
```

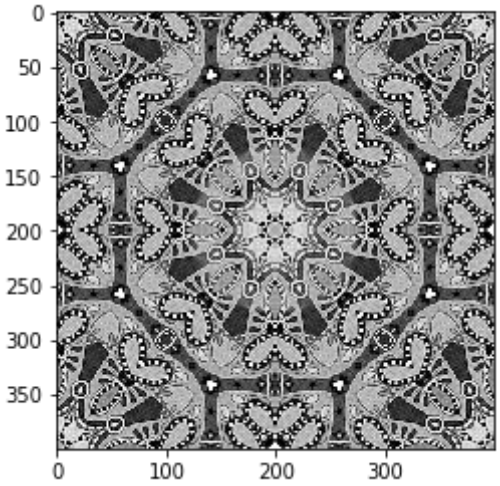
Part b

Find the pattern approximation using 100 largest eigenvalues/eigenvectors.

- Index into above variables to choose the first 100 eigenvalues and eigenvectors.
- You can use the command np.outer (http://docs.scipy.org/doc/numpy/reference/generated/numpy.outer.html) to find the outer product of two vectors

In [5]:
```
rank = 100
S = np.zeros(pattern.shape)
for i in range(rank):
    vec_i = eig_vectors[:,i]  # i-th largest eigenvector
    val_i = eig_vals[i]       # i-th largest eigenvalue
    S += val_i*np.outer(vec_i, vec_i, out=None)
plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[5]: `<matplotlib.image.AxesImage at 0x10dbcaf10>`

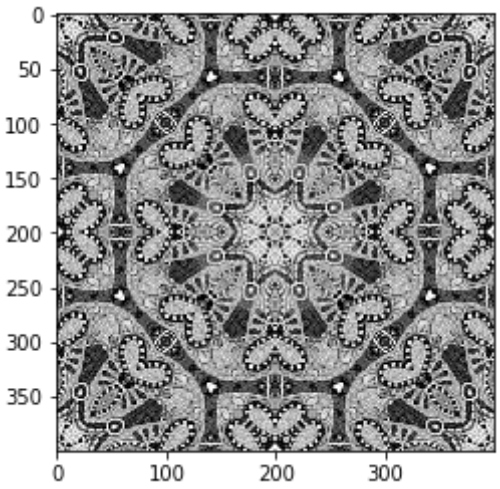


Part c

Find the pattern approximation using 50 largest eigenvalues/eigenvectors

In [6]:
```
rank = 50
S = np.zeros(pattern.shape)
for i in range(rank):
    vec_i = eig_vectors[:,i]  # i-th largest eigenvector
    val_i = eig_vals[i]       # i-th largest eigenvalue
    S += val_i*np.outer(vec_i, vec_i, out=None)
plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[6]: <matplotlib.image.AxesImage at 0x10dc1f6d0>



Now try decreasing the amount of eigenvalues/eigenvectors used in the pattern approximation. At what point does the image, start to substantially look different?

In [8]:
```
rank = 20
S = np.zeros(pattern.shape)
for i in range(rank):
    vec_i = eig_vectors[:,i]  # i-th largest eigenvector
    val_i = eig_vals[i]       # i-th largest eigenvalue
    S += val_i*np.outer(vec_i, vec_i, out=None)
plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[8]: <matplotlib.image.AxesImage at 0x10de769d0>