EE16A Homework 4

Image Compression

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

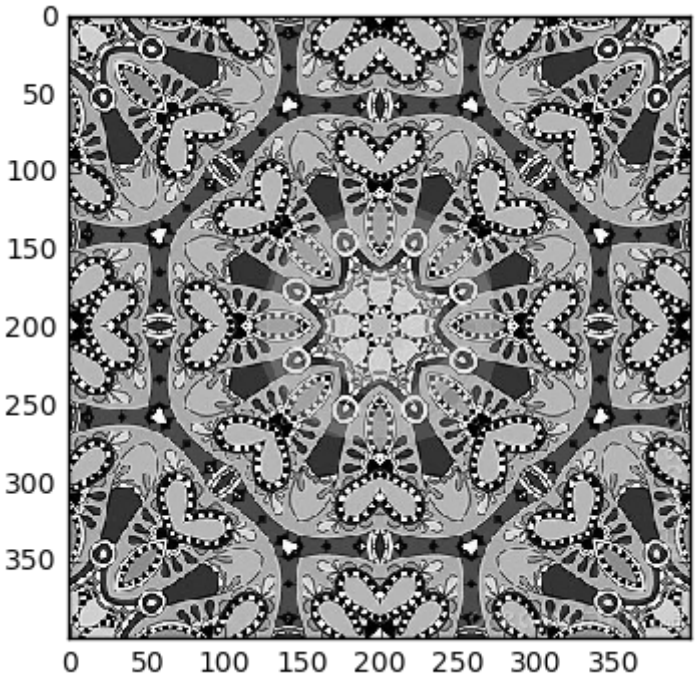
```
Populating the interactive namespace from numpy and matplotlib
```

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

Part b

In [23]:
```
#Load Pattern Image
pattern = np.load('pattern.npy')
plt.imshow(pattern, cmap='gray', interpolation='nearest')
```

Out[23]:
```
<matplotlib.image.AxesImage at 0x110b27eb8>
```



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 [28]:
```
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]
print(np.shape(pattern))
```

```
(400, 400)
```

Part c

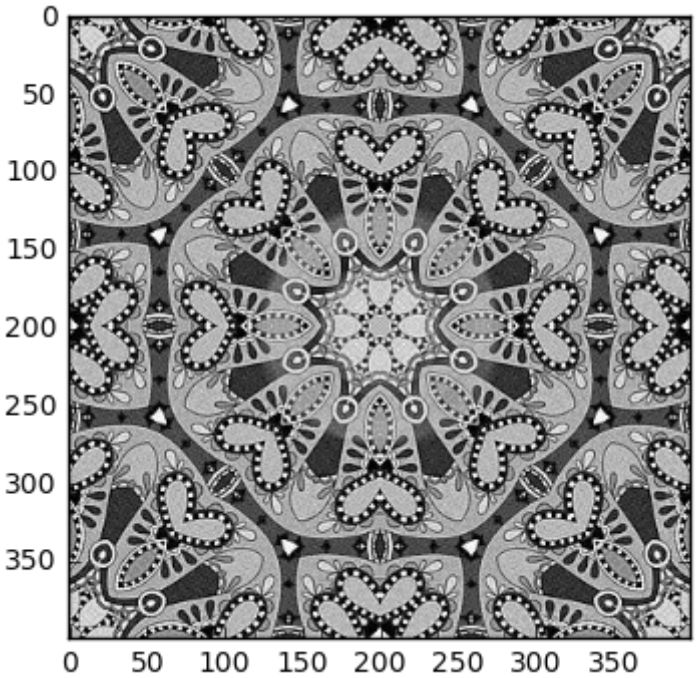
Find the pattern approximation using 100 largest eigenvalues/eigenvectors.

- 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 [29]:
```
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 += np.outer(np.outer(vec_i, val_i), vec_i.T)

plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[29]:
```
<matplotlib.image.AxesImage at 0x111c552e8>
```



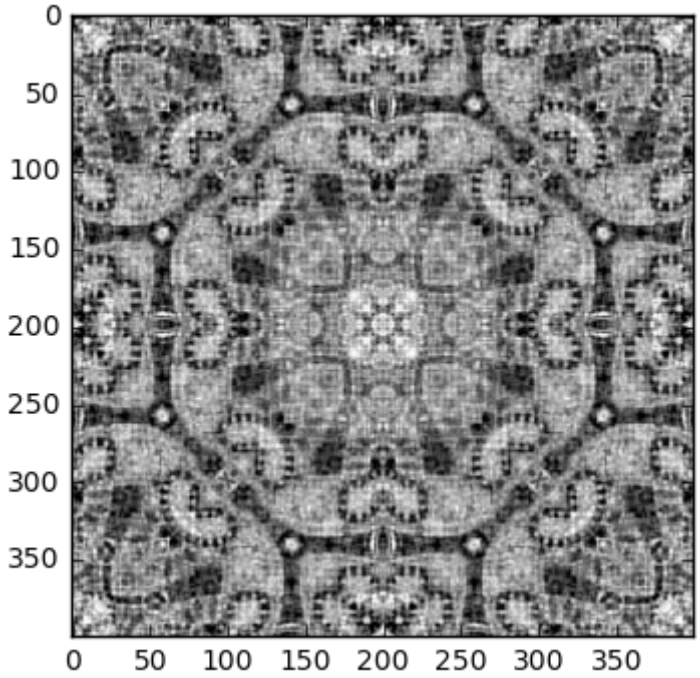
Part d

Find the pattern approximation using 50 largest eigenvalues/eigenvectors

In [36]:
```
rank = 15
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 += np.outer(np.outer(vec_i, val_i), vec_i.T)

plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[36]: <matplotlib.image.AxesImage at 0x1171ddc50>



Paths of a Surfer

In [22]:
```
# There is no required ipython component, but you may wish to use iPython fo
B = np.array([[0, 1, 1/3, 1/3],
              [0, 0, 1/3, 1/3],
              [0, 0, 0, 1/3],
              [1, 0, 1/3, 0]])
vbi = np.array([1/4, 1/4, 1/4, 1/4])
print(np.dot(B, np.dot(B, np.dot(B, np.dot(B, np.dot(B, np.dot(B, np.dot(B,

C = np.array([[0, 1, 0, 0, 0],
             [1, 0, 0, 0, 0],
             [0, 0, 0, 1, 1],
             [0, 0, 1/2, 0, 0],
             [0, 0, 1/2, 0, 0]])
vci = np.array([1/5, 1/5, 1/5, 1/5, 1/5])
final = np.dot(C, vci)
for _ in range(100):
    final = np.dot(C, final)
print(final)
```

```
[ 0.33373554  0.1663576   0.12448983  0.37541703]
[ 0.2  0.2  0.4  0.1  0.1]
```

In [ ]: