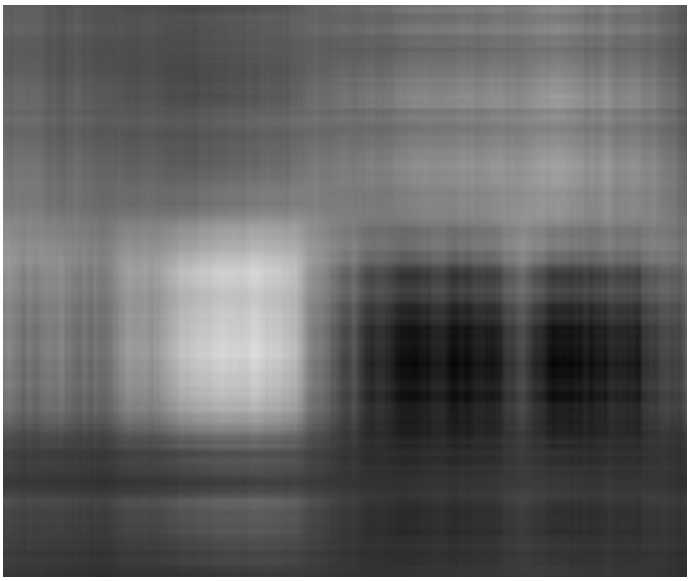
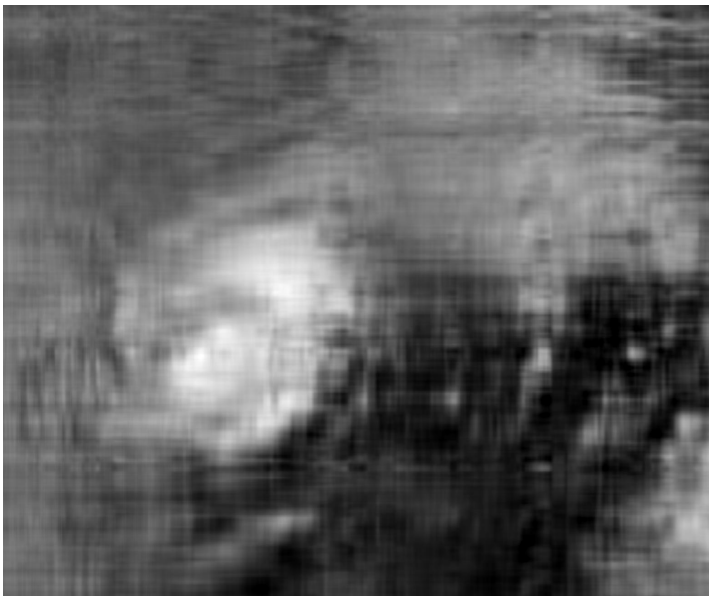
5. Programming

(a) After performing SVD on this matrix and zero out all but top k(2,10,40) singular values to form an approximation. Result images are shown as below:

- k = 2



- k = 10



- K = 40

result = [0.28258451691929531, 0.15928758406875393, 0.084079335040156966]

when k = 2, result = 0.28258451691929531
when k = 10, result = 0.15928758406875393
when k = 40, result = 0. 084079335040156966

(b) How many numbers do you need to describe the approximation

Result = [5690, 28450, 113800]
when k = 2, result = 5690
when k = 10, result = 28450
when k = 40, result = 113800

My code is shown as below:

```
import numpy as np
from PIL import Image


################################### Part A #######################################
# Load Image
image = Image.open('harvey-saturday-goes7am.jpg')
grey = image.convert("L")

# Image to array
X = np.asarray(grey)
```

```
# k value
k_list = [2,10,40]
result_list = []
partb_list = []

for k in k_list:
    # SVD
    U,s,Vt = np.linalg.svd(X, full_matrices = False)
    s[k:] = 0
    S = np.diag(s)
    X_app = np.dot(np.dot(U,S), Vt)

    # Show and save approximate_image
    img = Image.fromarray(X_app)
    if(img.mode != 'RGB'):
        img = img.convert('RGB')

    img.save(str(k)+'.jpg')

    # calculate ||X-X_app||f / ||X||f
    temp = X - X_app
    result = np.linalg.norm(temp,'fro') / np.linalg.norm(X,'fro')
    result_list.append(result)

################################### Part B ########################################
    # How many numbers do you need to describe the approximation
    left = U[:,:k]
    left_row = left.shape[0]
    left_column = left.shape[1]
    num_left = left_row * left_column

    right = Vt[:k,:]
    right_row = right.shape[0]
    right_column = right.shape[1]
    num_right = right_row * right_column

    partb = num_left + num_right + k
    partb_list.append(partb)
```