hw3_computing_1

April 14, 2020

1 HW3 Computing Problem 1

[1]:
```
import scipy as spy
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.io import loadmat
import math
import pandas as pd
```

[2]:
```
faces1 = loadmat('./HW3data/faces1.mat')
```

[3]:
```
faces1=faces1['Y1']
```

[4]:
```
faces1=faces1.astype(np.float32)
```

[5]:
```
faces1.shape
```

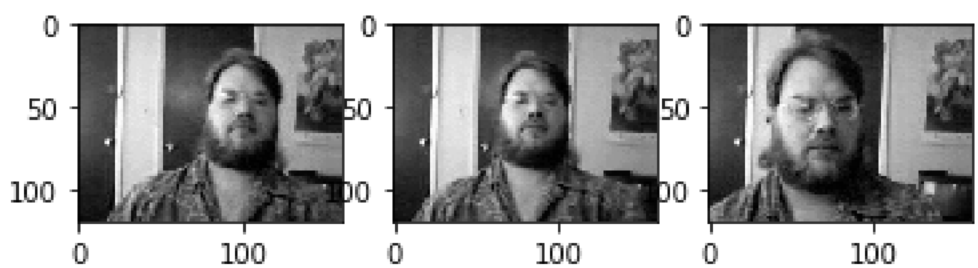
[5]:
```
(19200, 109)
```

[6]:
```
120*160
```

[6]:
```
19200
```

[7]:
```
fig, ax = plt.subplots(1, 3)
ax[0].imshow(faces1[:,0].reshape(160,120).T,cmap='gray')
ax[1].imshow(faces1[:,5].reshape(160,120).T,cmap='gray')
ax[2].imshow(faces1[:,108].reshape(160,120).T,cmap='gray')
```

[7]:
```
<matplotlib.image.AxesImage at 0xb2164ed68>
```

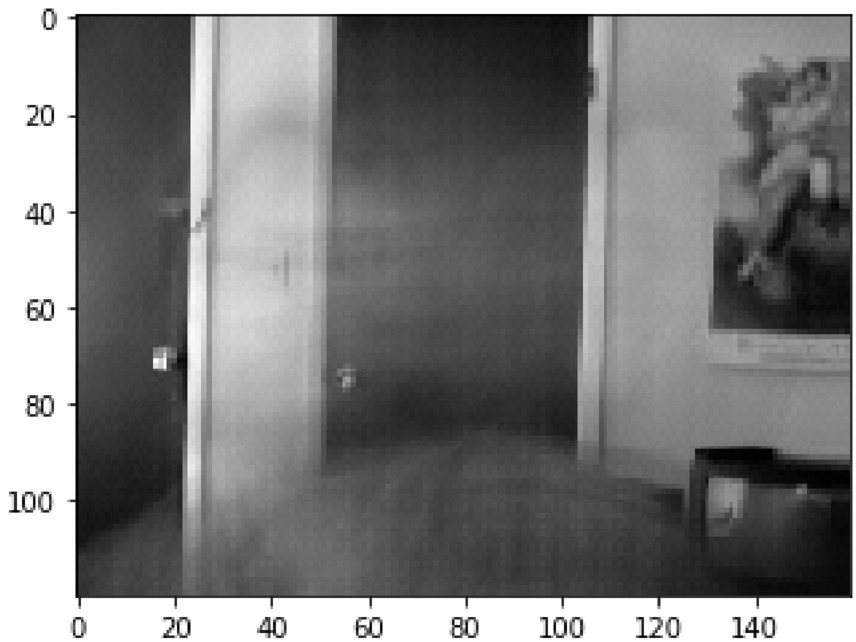


1.0.1 (a) Ensemble average image

[8]:
```
mu=np.mean(faces1,axis=1)
```

[9]:
```
plt.imshow(mu.reshape(160,120).T,cmap='gray')
```

[9]:
```
<matplotlib.image.AxesImage at 0xb2178a940>
```



1.0.2 (b) mean subtracted image

[10]:
```
mean_intensity=np.mean(faces1,axis=0)
```

[11]:
```
mean_intensity.shape
```

[11]:
```
(109,)
```

[12]:
```
faces1[:,0].mean()
```

[12]:
```
105.785675
```

[13]:
```
mean_intensity.shape
```

[13]:
```
(109,)
```

[14]:
```
faces1.shape
```

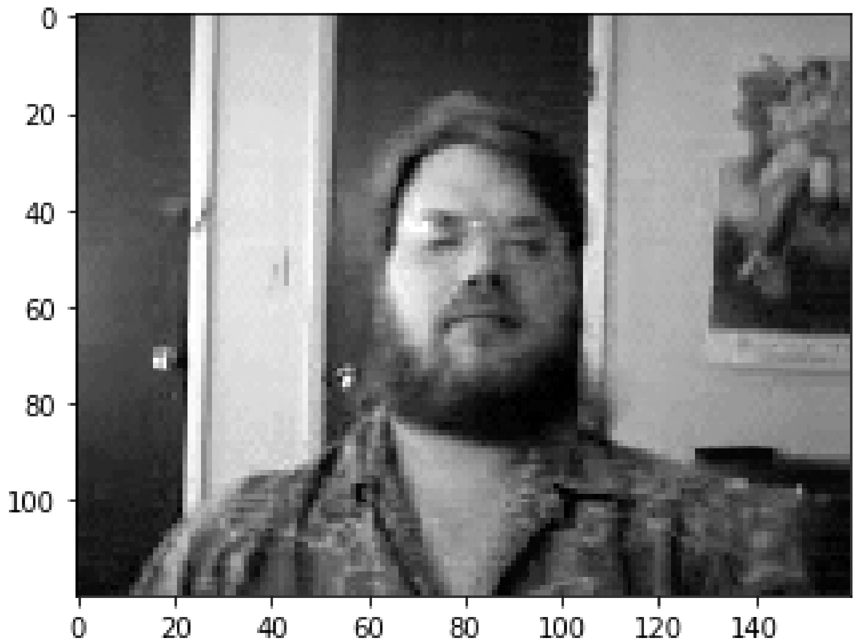
[14]:
```
(19200, 109)
```

[15]:
```python
mean_sub=faces1 - mean_intensity
```

Original

[16]:
```python
plt.imshow(faces1[:,6].reshape(160,120).T,cmap='gray')
```

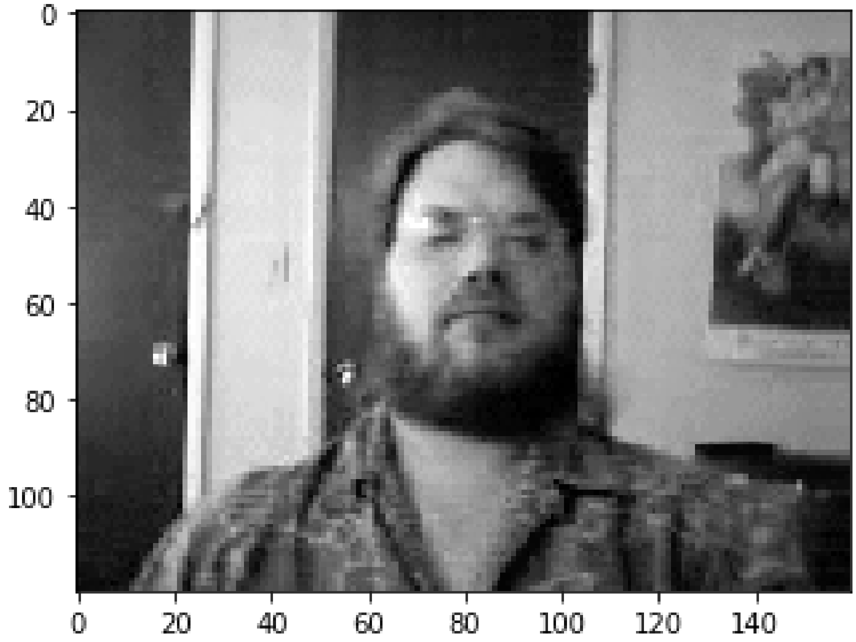
[16]: <matplotlib.image.AxesImage at 0xb216a3ef0>



Mean subtracted

[17]:
```python
plt.imshow(mean_sub[:,6].reshape(160,120).T,cmap='gray')
```

[17]: <matplotlib.image.AxesImage at 0xb221d6f98>



```
[18]: mean_sub.shape
```

```
[18]: (19200, 109)
```

1.0.3 (c) Determining eigenpictures by snapshot method

```
[19]: Ct=np.matmul(mean_sub.T,mean_sub)
```

```
[21]: eigvals, eigvecs = np.linalg.eigh(Ct)
```

```
[23]: eigvecs=np.flip(eigvecs,axis=1)
```

```
[24]: eigvecs.shape
```

```
[24]: (109, 109)
```

```
[25]: eigvals.shape
```

```
[25]: (109,)
```

```
[26]: eigvals=np.abs(np.round(eigvals,2))
```

```
[27]: eigvals=np.flip(eigvals)
```

```
[28]: sigmas = np.sqrt(eigvals)
```

```
[29]: Sigma = np.diag(sigmas)
```

```
[30]: Sigma.shape
```

```
[30]: (109, 109)
```

[33]:
```
Sigma_plus = np.linalg.pinv(Sigma)
```

[34]:
```
U = np.matmul(np.matmul(mean_sub,eigvecs),Sigma_plus)
```

[36]:
```
mean_sub.shape,eigvecs.shape,Sigma_plus.shape
```

[36]:
```
((19200, 109), (109, 109), (109, 109))
```

[37]:
```
U.shape
```

[37]:
```
(19200, 109)
```

[38]:
```
np.round(np.dot(U[:,0],U[:,1]))
```

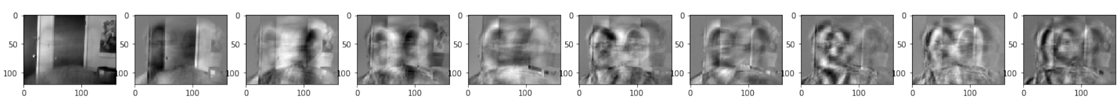
[38]:
```
-0.0
```

[39]:
```
np.round(np.dot(U[:,10],U[:,10]))
```

[39]:
```
1.0
```

Eigenpictures

[120]:
```
fig, ax = plt.subplots(1, 10,figsize=(25,25))
for i in range(10):
    ax[i].imshow(U[:,i].reshape(160,120).T,cmap='gray')
```



veryfying with SVD

[41]:
```
U,S,V=np.linalg.svd(mean_sub,full_matrices=False)
```

[42]:
```
U.shape
```

[42]:
```
(19200, 109)
```

[43]:
```
V.shape
```

[43]:
```
(109, 109)
```

[44]:
```
k=10
```

[45]:
```
A_k = np.dot(U[:,:k],np.dot(np.diag(S[:k]),V[:k,:]))
```

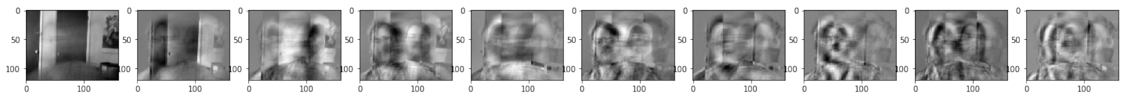
[46]:
```
A_k.shape
```

[46]:
```
(19200, 109)
```

[47]:
```
np.dot(U[:,0],U[:,0])
```

[47]:
```
1.0
```

[48]:
```
fig, ax = plt.subplots(1, 10,figsize=(25,25))
for i in range(10):
    ax[i].imshow(U[:,i].reshape(160,120).T,cmap='gray')
```



[167]:
```
mean_sub[:,6].shape
```

[167]: (19200,)

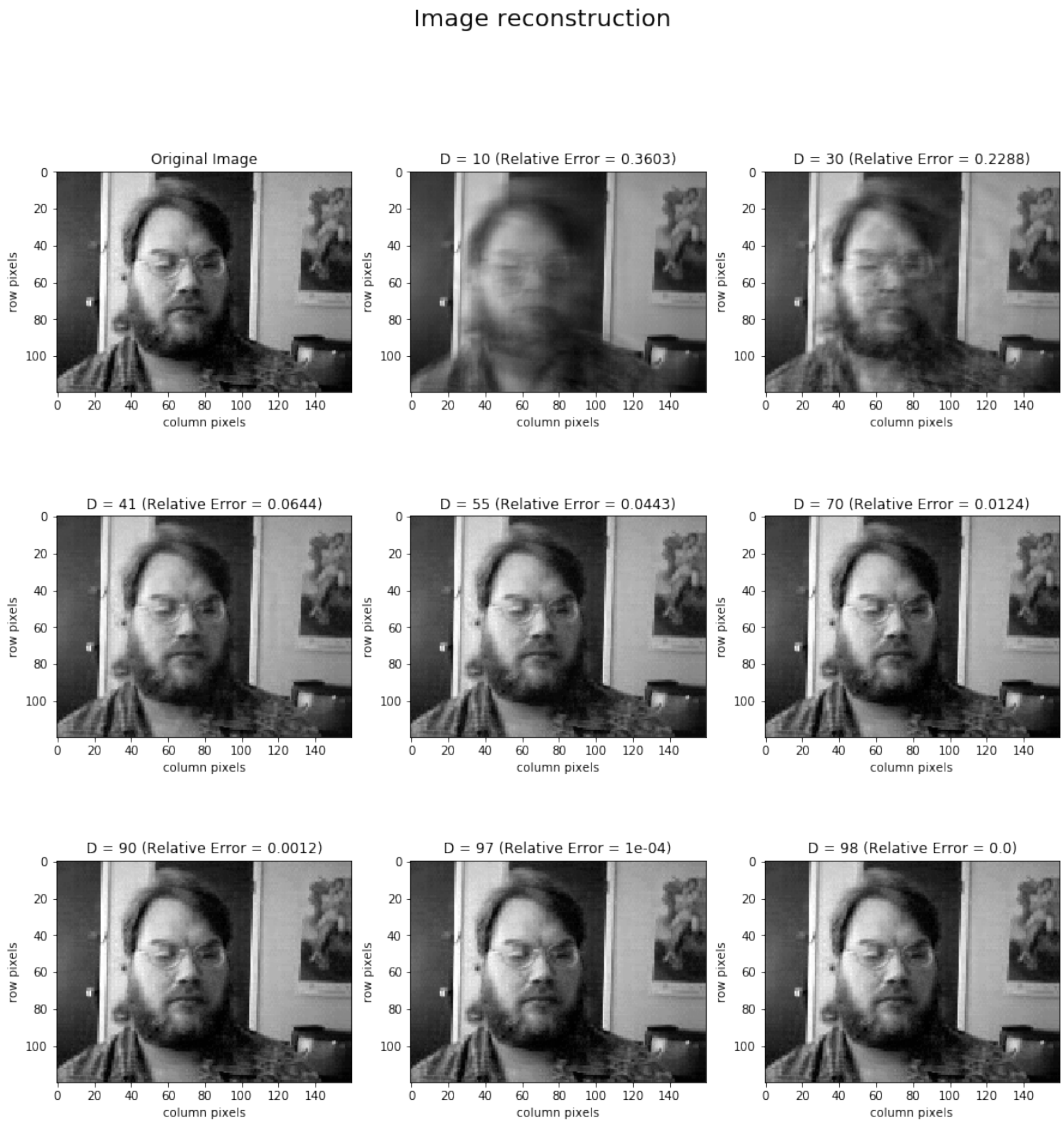
[228]:
```
U.shape
```

[228]: (19200, 109)

1.0.4 (d) Partial reconstruction

[195]:
```
x=mean_sub[:,108]
fig, ax = plt.subplots(3, 3,figsize=(15,15))
fig.suptitle('Image reconstruction', fontsize=20)
ax[0,0].set_title('Original Image')
ax[0,0].set_xlabel('column pixels')
ax[0,0].set_ylabel('row pixels')
ax[0,0].imshow(x.reshape(160,120).T,cmap='gray')
D_list=[10,30,41,55,70,90,97,98]
ctr=0
for row in range(3):
    for col in range(3):
        if (row==0 and col==0):
            continue
        else:
            xD=np.matmul(np.matmul(U[:,0:D_list[ctr]],U[:,0:D_list[ctr]].T),x)
            relative_error=np.round(np.linalg.norm(x-xD)/np.linalg.norm(x),4)
            ax[row,col].set_title('D = '+str(D_list[ctr])+' (Relative Error =␣
 ↪'+str(relative_error)+')')
            ax[row,col].set_xlabel('column pixels')
            ax[row,col].set_ylabel('row pixels')
            ax[row,col].imshow(xD.reshape(160,120).T,cmap='gray')
            ctr +=1
```

0 0



[197]:
```
np.linalg.matrix_rank(faces1)
```

[197]: 98

1.0.5 (e) Graph of λ_i/λ_{max}

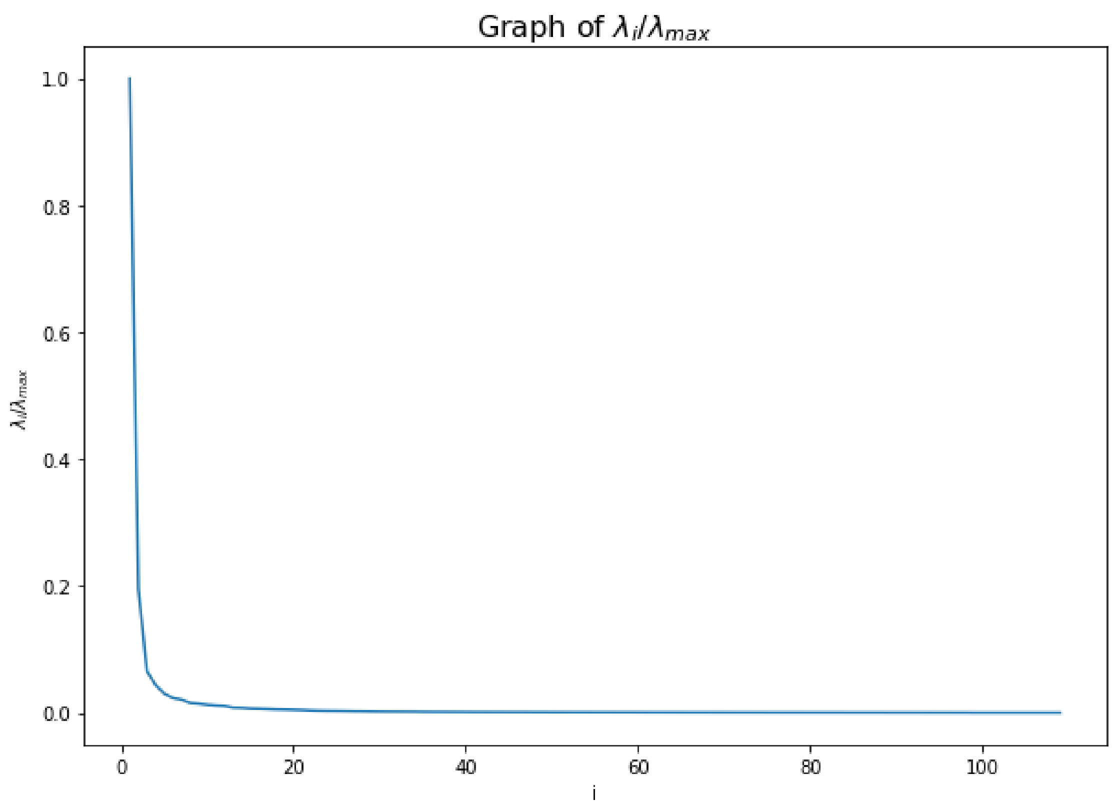
[184]:
```
lambdas=pd.DataFrame(eigvals,columns=['lambda'])
```

[185]:
```
lambdas['i'] = lambdas.index + 1
```

[186]:
```
lambdas['lambda_i/lambda_max'] = lambdas['lambda']/lambdas['lambda'].max()
```

[187]:
```
fig, ax = plt.subplots(1, 1,figsize=(10,7))
ax.set_title('Graph of $\lambda_i/\lambda_{max}$',fontsize=16)
ax.set_xlabel('i')
ax.set_ylabel('$\lambda_i/\lambda_{max}$')
ax.plot(lambdas['i'],lambdas['lambda_i/lambda_max'])
```

[187]:
```
[<matplotlib.lines.Line2D at 0xb1a40feb8>]
```



[188]:
```
lambdas
```

[188]:

	lambda	i	lambda_i/lambda_max
0	5.595008e+09	1	1.000000
1	1.091896e+09	2	0.195155
2	3.658350e+08	3	0.065386
3	2.449413e+08	4	0.043779
4	1.691472e+08	5	0.030232
5	1.309923e+08	6	0.023412
6	1.164114e+08	7	0.020806
7	8.676419e+07	8	0.015507
8	8.069685e+07	9	0.014423
9	7.049070e+07	10	0.012599
10	6.401132e+07	11	0.011441
11	6.071448e+07	12	0.010852

12	4.550784e+07	13	0.008134
13	4.317619e+07	14	0.007717
14	3.798976e+07	15	0.006790
15	3.584921e+07	16	0.006407
16	3.329347e+07	17	0.005951
17	3.011722e+07	18	0.005383
18	2.881656e+07	19	0.005150
19	2.619385e+07	20	0.004682
20	2.384495e+07	21	0.004262
21	1.998509e+07	22	0.003572
22	1.779970e+07	23	0.003181
23	1.675465e+07	24	0.002995
24	1.604728e+07	25	0.002868
25	1.523827e+07	26	0.002724
26	1.386105e+07	27	0.002477
27	1.323308e+07	28	0.002365
28	1.179135e+07	29	0.002107
29	1.095026e+07	30	0.001957
..	...	...	...
79	1.494306e+06	80	0.000267
80	1.456422e+06	81	0.000260
81	1.426809e+06	82	0.000255
82	1.402486e+06	83	0.000251
83	1.347631e+06	84	0.000241
84	1.325386e+06	85	0.000237
85	1.286731e+06	86	0.000230
86	1.249597e+06	87	0.000223
87	1.183466e+06	88	0.000212
88	1.032372e+06	89	0.000185
89	1.009450e+06	90	0.000180
90	9.937971e+05	91	0.000178
91	9.416671e+05	92	0.000168
92	8.996682e+05	93	0.000161
93	8.347216e+05	94	0.000149
94	8.178806e+05	95	0.000146
95	7.828898e+05	96	0.000140
96	7.305701e+05	97	0.000131
97	6.980302e+05	98	0.000125
98	0.000000e+00	99	0.000000
99	0.000000e+00	100	0.000000
100	0.000000e+00	101	0.000000
101	0.000000e+00	102	0.000000
102	0.000000e+00	103	0.000000
103	0.000000e+00	104	0.000000
104	0.000000e+00	105	0.000000
105	0.000000e+00	106	0.000000
106	0.000000e+00	107	0.000000

```
107  0.000000e+00  108                0.000000
108  0.000000e+00  109                0.000000

[109 rows x 3 columns]
```

1.0.6 (f) Classification algorithm using PCA

[41]:
```
#### see report
```