CMSE 820: Homework #4

Due on October 6, 2019 at 11:59pm

Professor Yuying Xie

Boyao Zhu

Problem 1

Solution

To find the second principal component, $\mathbf{u}_2$, we use the fact that $\mathbf{u}_1^T\mathbf{x}$ and $\mathbf{u}_2^T\mathbf{x}$ need to be uncorrelated. This implies that $\mathbf{u}_2$ is orthogonal to $\mathbf{u}_1$. Indeed from

$$\mathbb{E}[(\mathbf{u}_1^T\mathbf{x})(\mathbf{u}_2^T\mathbf{x})] = [\mathbf{u}_1^T\mathbf{x}\mathbf{x}^T\mathbf{u}_2] = \mathbf{u}_1^T\Sigma_x\mathbf{u}_2 = \lambda\mathbf{u}_1^T\mathbf{u}_2 = 0$$

and $\lambda_1 \neq 0$, we have $\mathbf{u}_1^T\mathbf{u}_2 = 0$. Thus, to find $\mathbf{u}_2$, we need to solve the following optimization problem:

$$\max_{\mathbf{u}_2\in\mathbb{R}^D} \mathbf{u}_2^T\Sigma_x\mathbf{u}_2 \qquad \text{s.t.} \qquad \mathbf{u}_2^T\mathbf{u}_2 = 1 \qquad \mathbf{u}_1^T\mathbf{u}_2 = 0$$

we define the Lagrangian

$$\mathcal{L} = \mathbf{u}_2^T\Sigma_x\mathbf{u}_2 + \lambda_2(1-\mathbf{u}_2^T\mathbf{u}_2) + \gamma\mathbf{u}_1^T\mathbf{u}_2$$

The necessary conditions for $(\mathbf{u}_2, \lambda_2, \gamma)$ to be an extremum are

$$\Sigma_x\mathbf{u}_2 + \frac{\gamma}{2}\mathbf{u}_1 = \lambda_2\mathbf{u}_2, \qquad \text{s.t.} \qquad \mathbf{u}_2^T\mathbf{u}_2 = 1 \qquad \mathbf{u}_1^T\mathbf{u}_2 = 0$$

from which it follows that $\mathbf{u}_1^T\Sigma_x\mathbf{u}_2 + \frac{\gamma}{2}\mathbf{u}_1^T\mathbf{u}_1 = \lambda_1\mathbf{u}_1^T\mathbf{u}_2 + \frac{\gamma}{2} = \lambda_2\mathbf{u}_1^T\mathbf{u}_2$, and so $\gamma = 2(\lambda_2-\lambda_1)\mathbf{u}_1^T\mathbf{u}_2 = 0$. This implies that $\Sigma_x\mathbf{u}_2 = \lambda_2\mathbf{u}_2$ and that the extremum value is $\mathbf{u}_2^T\Sigma_x\mathbf{u}_2 = \lambda_2 = \mathrm{Var}(y_2)$. Therefore, $\mathbf{u}_2$ is the leading eigenvector of Σ_x restricted to the orthogonal complement of $\mathbf{u}_1$. Since the eigenvalues of Σ_x are distinct, $\mathbf{u}_2$ is the eigenvector of Σ_x associated with its second-largest eigenvalue.

To find the remaining principal components, we use that fact that for all for $i \neq j, y_i = \mathbf{u}_i^T\mathbf{x}$ and $y_i = \mathbf{u}_j^T\mathbf{x}$ need to be uncorrelated, hence

$$\mathrm{Var}(y_iy_j) = \mathbb{E}[\mathbf{u}_i^T\mathbf{x}\mathbf{x}^T\mathbf{u}_j] = \mathbf{u}_i^T\Sigma_x\mathbf{u}_j = 0$$

Using induction, assume that $\mathbf{u}_1, \cdots, \mathbf{u}_{i-1}$ are the unit-length eigenvectors of Σ_x associated with its top $i-1$ eigenvalues, and let $\mathbf{u}_i$ be the vector defining the ith principal component, y_i. Then, $\Sigma_x\mathbf{u}_j = \lambda_j\mathbf{u}_j$ for $j = 1, \cdots, i-1$ and $\mathbf{u}_i^T\Sigma_x\mathbf{u}_j = \lambda_j\mathbf{u}_i^T\mathbf{u}_j = 0$ for all $j = 1, \cdots, i-1$. Since $\lambda_j > 0$, we have that $\mathbf{u}_i^T\mathbf{u}_j = 0$ for all $j = 1, \cdots, i-1$. To compute $\mathbf{u}_i$, we build the Lagrangian

$$\mathcal{L} = \mathbf{u}_i^T\Sigma_x\mathbf{u}_i + \lambda_i(1-\mathbf{u}_i^T\mathbf{u}_i) + \sum_{j=1}^{i-1}\gamma_i\mathbf{u}_i^T\mathbf{u}_j$$

The necessary condition for $(\mathbf{u}_i, \lambda_i, \gamma_1, \cdots, \gamma_{j-1})$ to be an extremum are

$$\Sigma_x\mathbf{u}_i + \sum_{j=1}^{i-1}\frac{\gamma_j}{2}\mathbf{u}_j = \lambda_i\mathbf{u}_i, \qquad \mathbf{u}_i^T\mathbf{u}_i = 1 \quad \text{and} \quad \mathbf{u}_i^T\mathbf{u}_j = 0, \quad j = 1, \cdots, i-1$$

from which it follows that for all $j = 1, \cdots, i-1$, we have $\mathbf{u}_j^T\Sigma_x\mathbf{u}_i + \frac{\gamma_j}{2} = \lambda_j\mathbf{u}_j^T\mathbf{u}_i + \frac{\gamma_j}{2} = \lambda_i\mathbf{u}_j^T\mathbf{u}_i$, and so $\gamma_j = 2(\lambda_j - \lambda_i)\mathbf{u}_j^T\mathbf{u}_i = 0$. Since the associated extremum value is $\mathbf{u}_i^T\Sigma_x\mathbf{u}_i = \lambda_i = \lambda_i = \mathrm{Var}(y_i), \mathbf{u}_i$ is the leading eigenvector of Σ_x restricted to the orthogonal complement of the span of $\mathbf{u}_1, \cdots, \mathbf{u}_{i-1}$. Since the eigenvalues of $Sigma_x$ are distinct, $\mathbf{u}_i$ is the eigenvector of Σ_x associated with the ith-largest eigenvalue. Therefore, when the eigenvalues of Σ_x are distinct, each eigenvector $\mathbf{u}_i$ is unique, and hence so are the principal components of $\mathbf{x}$.

Problem 2

Solution

a.

Since A is a symmetric real matrix $\in \mathbb{R}^{n\times n}$, its eigenvalues are real and its eigenvectors form a basis of $\mathbb{R}^D$. Moreover, the eigenvectors are unique, and the eigenvectors corresponding to different eigenvalues are orthogonal to each other. Since the problem is to prove maximum of $v^T Av = \lambda_1$, the largest eigenvalue of A, which is a constrained optimization problem. we could use the method of Lagrange multipliers with constraint, $\|v\|_2^2 = v^T v = 1$.

$$\mathcal{L} = v^T Av + \lambda(1 - v^T v)$$

where $\lambda \in \mathbb{R}$ is the Lagrange multiplier. From computing the derivatives of $\mathcal{L}$ with respect to (v, λ) and setting them to zero, we obtain the following necessary condition for (v, λ) to be an extremum of $\mathcal{L}$.

$$Av = \lambda v \qquad \text{and} \qquad v^T v = 1$$

This means that v is an eigenvector of A with associated eigenvalue λ. Since the extremum is maximum, the optimal solution for v is given by the eigenvector of A associated with its largest eigenvalue $\lambda = \lambda_1$.

b.

Similarly, here the extremum is minimum, the optimal solution for v is given by the eigenvector of A associated with its smallest eigenvalue $\lambda = \lambda_n$.

Problem 3

Solution

For the first inequility:

$$\begin{aligned}
\lambda_k(A+E) &= \max_{\dim(V)=k} \min_{v\in V:\|v\|=1} v^T(A+E)v \\
&= \max_{\dim(V)=k} \min_{v\in V:\|v\|=1} v^T Av + v^T Ev \\
&\geq \max_{\dim(V)=k} (\min_{v\in V:\|v\|=1} v^T Av + \min_{v\in V:\|v\|=1} v^T Ev) \\
&\geq \max_{\dim(V)=k} \lambda_n(A) + \lambda_n(E) \\
&\geq \lambda_k(A) + \lambda_k(E) \\
&\geq \lambda_k(A) + \lambda_n(E)
\end{aligned}$$

For the second inequility:

$$\begin{aligned}
\lambda_k(A+E) &= \min_{\dim(V)=n-k+1} \max_{v\in V:\|v\|=1} v^T(A+E)v \\
&= \min_{\dim(V)=n-k+1} \max_{v\in V:\|v\|=1} v^T Av + v^T Ev \\
&\leq \min_{\dim(V)=n-k+1} (\max_{v\in V:\|v\|=1} v^T Av + \max_{v\in V:\|v\|=1} v^T Ev) \\
&\leq \min_{\dim(V)=n-k+1} \lambda_1(A) + \lambda_1(E) \\
&\leq \lambda_1(A) + \lambda_1(E) \\
&\leq \lambda_k(A) + \lambda_1(E)
\end{aligned}$$

Problem 4

Solution All solutions are attached in the back. The plots are displayed from top to bottom are mean face, first eigenface and second eigenface. and faces of $\mu + a_i u_i$ with various values with a_i are displayed. The first row of each individual corresponds to the variation along u_1, while the second row corresponds to the variation along u_2.

For the plots (a), we can learn that the first eigenface u_1 differs from the mean face μ in terms of the lack of illumination. In some sense, they are almost the same. On the other hand, u_2 differs from μ in terms of it illumination from the right.

For the plots (b) we can clearly see that, for all three individuals, the first eigenface u_1 captures the information about the horizontal illumination, while the second eigenface u_2 captures that about the vertical illumination.

Untitled1

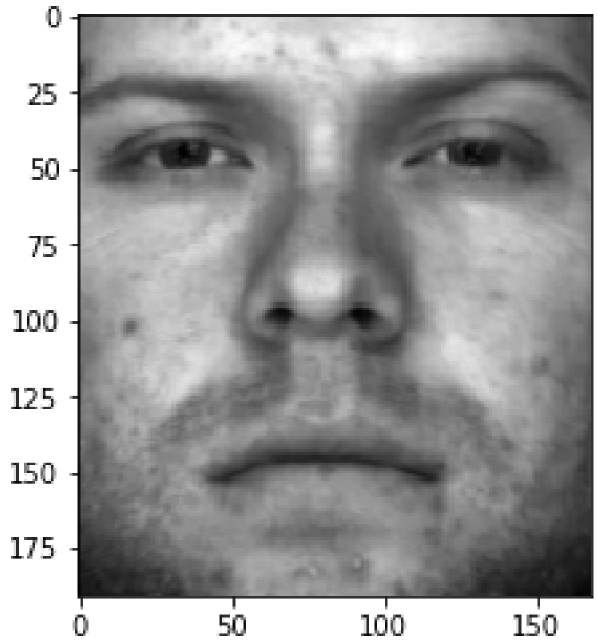
October 6, 2019

```
In [39]: import csv
         import pandas as pd
         import matplotlib.pyplot as plt
         import numpy as np

         X1 = pd.read_csv("M1.csv").as_matrix().reshape(-1).T
         X2 = pd.read_csv("M2.csv").as_matrix().reshape(-1).T
         X3 = pd.read_csv("M3.csv").as_matrix().reshape(-1).T
         X4 = pd.read_csv("M4.csv").as_matrix().reshape(-1).T
         X5 = pd.read_csv("M5.csv").as_matrix().reshape(-1).T
         X6 = pd.read_csv("M6.csv").as_matrix().reshape(-1).T
         X7 = pd.read_csv("M7.csv").as_matrix().reshape(-1).T
         X8 = pd.read_csv("M8.csv").as_matrix().reshape(-1).T
         X9 = pd.read_csv("M9.csv").as_matrix().reshape(-1).T
         X10= pd.read_csv("M10.csv").as_matrix().reshape(-1).T

         X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T

         X_mean = np.mean(X,axis=1)
         plt.imshow(X_mean.reshape(191,168),cmap="gray")
```

```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: FutureWarning: Method .as_matri

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7: FutureWarning: Method .as_matri
  import sys
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: FutureWarning: Method .as_matri

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: FutureWarning: Method .as_matri
  if __name__ == '__main__':
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: FutureWarning: Method .as_mat
  # Remove the CWD from sys.path while we load stuff.
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11: FutureWarning: Method .as_mat
  # This is added back by InteractiveShellApp.init_path()
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:12: FutureWarning: Method .as_mat
  if sys.path[0] == '':
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:13: FutureWarning: Method .as_mat
  del sys.path[0]
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:14: FutureWarning: Method .as_mat
```

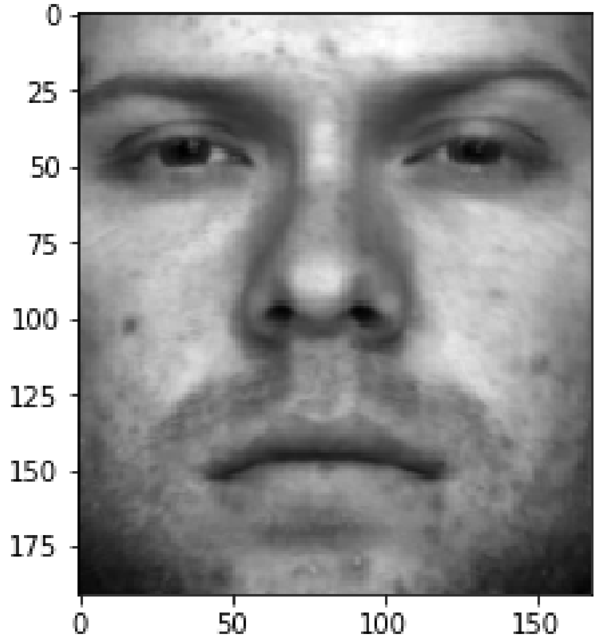
```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15: FutureWarning: Method .as_mat
  from ipykernel import kernelapp as app
```

Out[39]: <matplotlib.image.AxesImage at 0x118db9390>



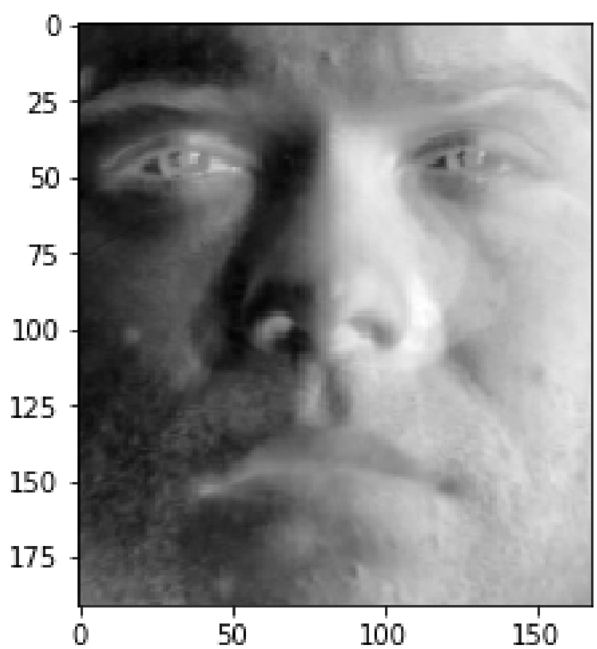
In [40]:
```
import numpy as np

mean_vec = np.mean(X, axis=0)
X = X-mean_vec
U,s,Vt = np.linalg.svd(X)

#first principal component
plt.imshow(U[:,0].reshape(191,168),cmap="gray")
```

Out[40]: <matplotlib.image.AxesImage at 0x4f1f57278>




```
In [41]: #second principal component
         plt.imshow(U[:,1].reshape(191,168),cmap="gray")
```

Out[41]: <matplotlib.image.AxesImage at 0x306a96940>



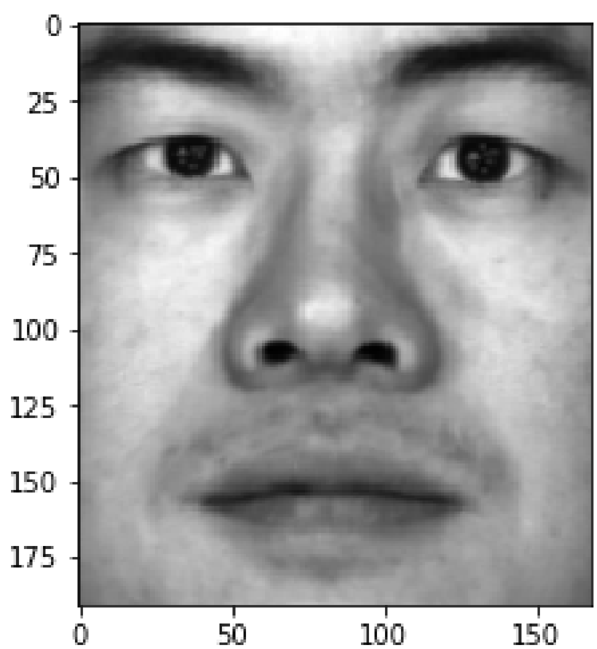
In [70]:
```
sigma = s[0:10] # standard deviation
a1 = np.array([-1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1])*sigma[0]
a2 = np.array([-1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1])*sigma[1]

fig = plt.figure(figsize=(13,13))

column = 11
row =1
for i in range(1,column*row+1):
    fig.add_subplot(row, column, i)
    plt.imshow(X_mean.reshape(191,168)+a1[i-1]*U[:,0].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()
fig = plt.figure(figsize=(13,13))
for i in range(1,column*row+1):
    fig.add_subplot(row, column, i)
    plt.imshow(X_mean.reshape(191,168)+a2[i-1]*U[:,1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()
```

In [71]:
```
X1 = pd.read_csv("H1.csv").as_matrix().reshape(-1).T
X2 = pd.read_csv("H2.csv").as_matrix().reshape(-1).T
X3 = pd.read_csv("H3.csv").as_matrix().reshape(-1).T
X4 = pd.read_csv("H4.csv").as_matrix().reshape(-1).T
X5 = pd.read_csv("H5.csv").as_matrix().reshape(-1).T
X6 = pd.read_csv("H6.csv").as_matrix().reshape(-1).T
X7 = pd.read_csv("H7.csv").as_matrix().reshape(-1).T
X8 = pd.read_csv("H8.csv").as_matrix().reshape(-1).T
X9 = pd.read_csv("H9.csv").as_matrix().reshape(-1).T
X10= pd.read_csv("H10.csv").as_matrix().reshape(-1).T

X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T
```

```
X_mean = np.mean(X,axis=1)
plt.imshow(X_mean.reshape(191,168),cmap="gray")
```

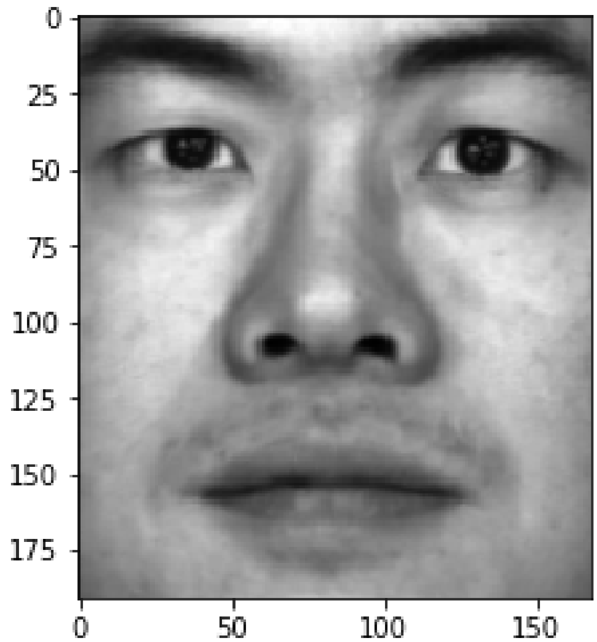
```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: Method .as_matri
  """Entry point for launching an IPython kernel.
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning: Method .as_matri

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: FutureWarning: Method .as_matri
  This is separate from the ipykernel package so we can avoid doing imports until
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: FutureWarning: Method .as_matri
  after removing the cwd from sys.path.
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: FutureWarning: Method .as_matri
  """
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: FutureWarning: Method .as_matri

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7: FutureWarning: Method .as_matri
  import sys
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: FutureWarning: Method .as_matri

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: FutureWarning: Method .as_matri
  if __name__ == '__main__':
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: FutureWarning: Method .as_mat
  # Remove the CWD from sys.path while we load stuff.
```

Out[71]: <matplotlib.image.AxesImage at 0x4f36c77b8>



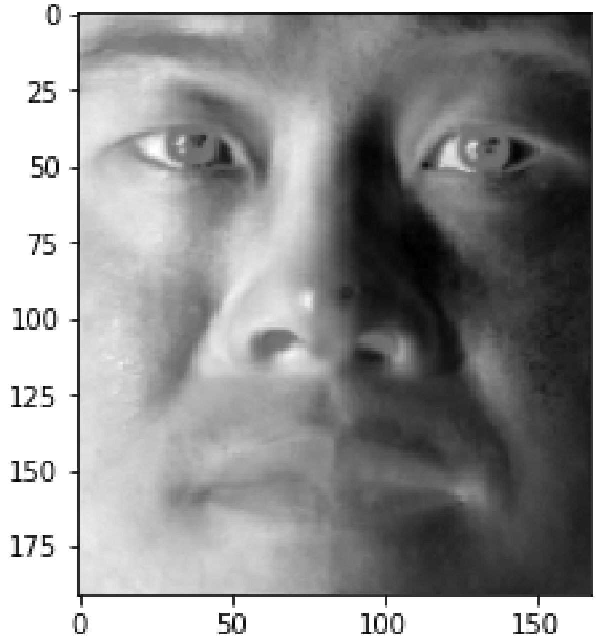
In [72]:
```
mean_vec = np.mean(X, axis=0)
X = X-mean_vec
U,s,Vt = np.linalg.svd(X)

#first principal component
plt.imshow(U[:,0].reshape(191,168),cmap="gray")
```

Out[72]: <matplotlib.image.AxesImage at 0x4f3fc6550>



In [73]:
```
#second principal component
plt.imshow(U[:,1].reshape(191,168),cmap="gray")
```

Out[73]: <matplotlib.image.AxesImage at 0x4f38b2940>

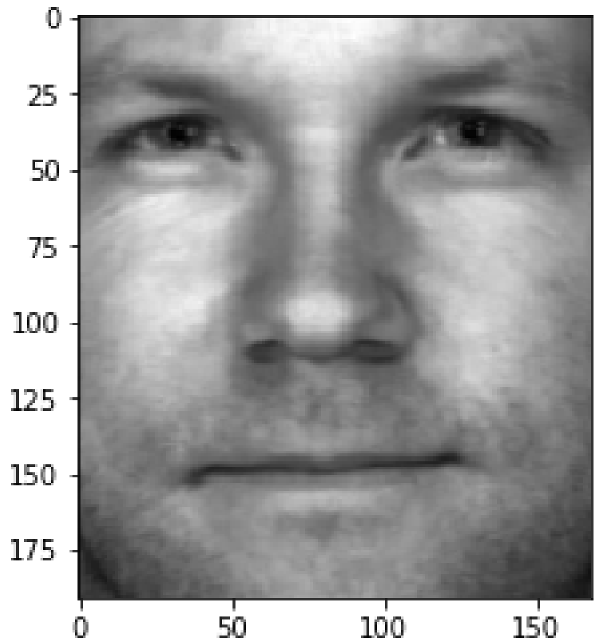


In [74]:
```
sigma = s[0:10] # standard deviation
a1 = np.array([-1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1])*sigma[0]
a2 = np.array([-1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1])*sigma[1]

fig = plt.figure(figsize=(13,13))

column = 11
row =1
for i in range(1,column*row+1):
    fig.add_subplot(row, column, i)
    plt.imshow(X_mean.reshape(191,168)+a1[i-1]*U[:,0].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()
fig = plt.figure(figsize=(13,13))
for i in range(1,column*row+1):
    fig.add_subplot(row, column, i)
    plt.imshow(X_mean.reshape(191,168)+a2[i-1]*U[:,1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()
```

```
In [75]: X1 = pd.read_csv("F1.csv").as_matrix().reshape(-1).T
         X2 = pd.read_csv("F2.csv").as_matrix().reshape(-1).T
         X3 = pd.read_csv("F3.csv").as_matrix().reshape(-1).T
         X4 = pd.read_csv("F4.csv").as_matrix().reshape(-1).T
         X5 = pd.read_csv("F5.csv").as_matrix().reshape(-1).T
         X6 = pd.read_csv("F6.csv").as_matrix().reshape(-1).T
         X7 = pd.read_csv("F7.csv").as_matrix().reshape(-1).T
         X8 = pd.read_csv("F8.csv").as_matrix().reshape(-1).T
         X9 = pd.read_csv("F9.csv").as_matrix().reshape(-1).T
         X10= pd.read_csv("F10.csv").as_matrix().reshape(-1).T

         X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T

         X_mean = np.mean(X,axis=1)
         plt.imshow(X_mean.reshape(191,168),cmap="gray")
```

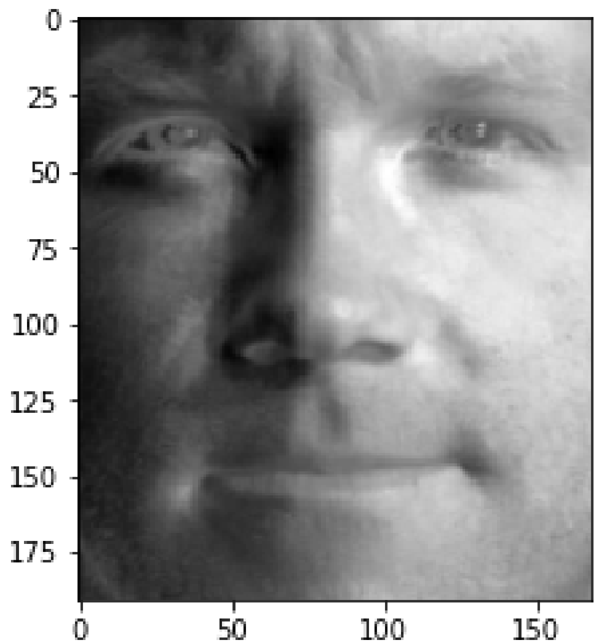
```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: Method .as_matr
  """Entry point for launching an IPython kernel.
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning: Method .as_matr

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: FutureWarning: Method .as_matr
  This is separate from the ipykernel package so we can avoid doing imports until
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: FutureWarning: Method .as_matr
  after removing the cwd from sys.path.
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: FutureWarning: Method .as_matr
  """
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: FutureWarning: Method .as_matr

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7: FutureWarning: Method .as_matr
  import sys
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: FutureWarning: Method .as_matr

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: FutureWarning: Method .as_matr
  if __name__ == '__main__':
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: FutureWarning: Method .as_mat
  # Remove the CWD from sys.path while we load stuff.
```

```
Out[75]: <matplotlib.image.AxesImage at 0x4f2c69358>
```



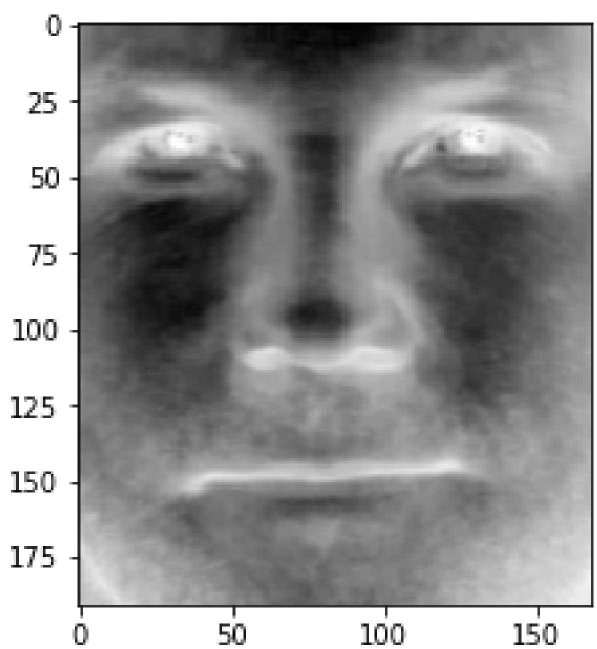
In [76]:
```
mean_vec = np.mean(X, axis=0)
X = X-mean_vec
U,s,Vt = np.linalg.svd(X)

#first principal component
plt.imshow(U[:,0].reshape(191,168),cmap="gray")
```

Out[76]: <matplotlib.image.AxesImage at 0x4f36eb630>



```
In [77]: #second principal component
         plt.imshow(U[:,1].reshape(191,168),cmap="gray")
```

Out[77]: <matplotlib.image.AxesImage at 0x4f36a99e8>



In [78]:
```
sigma = s[0:10] # standard deviation
a1 = np.array([-1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1])*sigma[0]
a2 = np.array([-1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1])*sigma[1]

fig = plt.figure(figsize=(13,13))

column = 11
row =1
for i in range(1,column*row+1):
    fig.add_subplot(row, column, i)
    plt.imshow(X_mean.reshape(191,168)+a1[i-1]*U[:,0].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()
fig = plt.figure(figsize=(13,13))
for i in range(1,column*row+1):
    fig.add_subplot(row, column, i)
    plt.imshow(X_mean.reshape(191,168)+a2[i-1]*U[:,1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()
```

In [ ]: