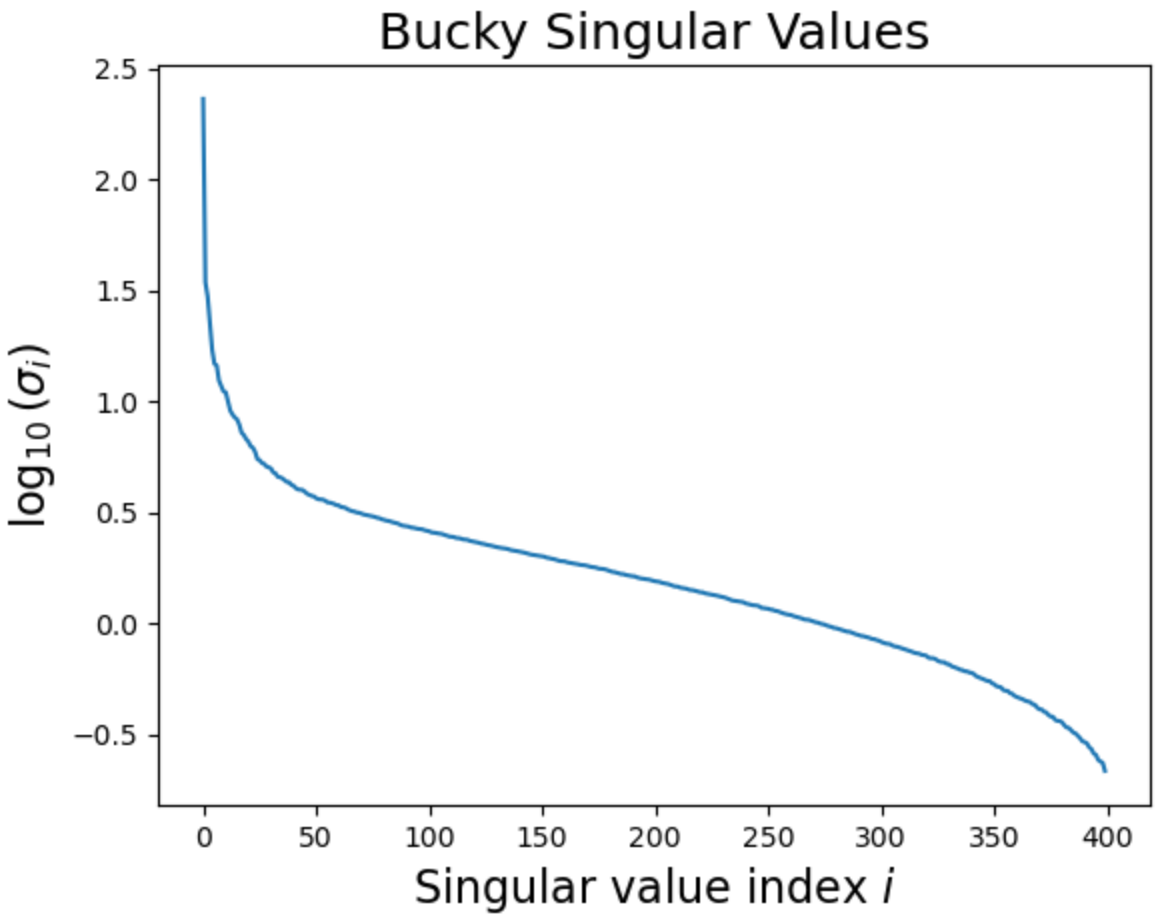
In [1]:
```
import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Load data for activity
#
in_data = loadmat('bucky.mat')
A = in_data['A']
##

# Load data for activity: Another option
# A = imageio.imread("Whateveryoulike.png")
# A = np.average(A[:,:,0:3], axis=2)/256

rows, cols = np.array(A.shape)
```

In [2]:
```
# Display image
fig = plt.figure()
ax = fig.add_subplot(111)

ax.imshow(A,cmap='gray')
ax.set_axis_off()
plt.show()
```


In [3]:
```
# Bucky's singular values

# Complete and uncomment line below
U, s, VT = np.linalg.svd(A, full_matrices=True)
```

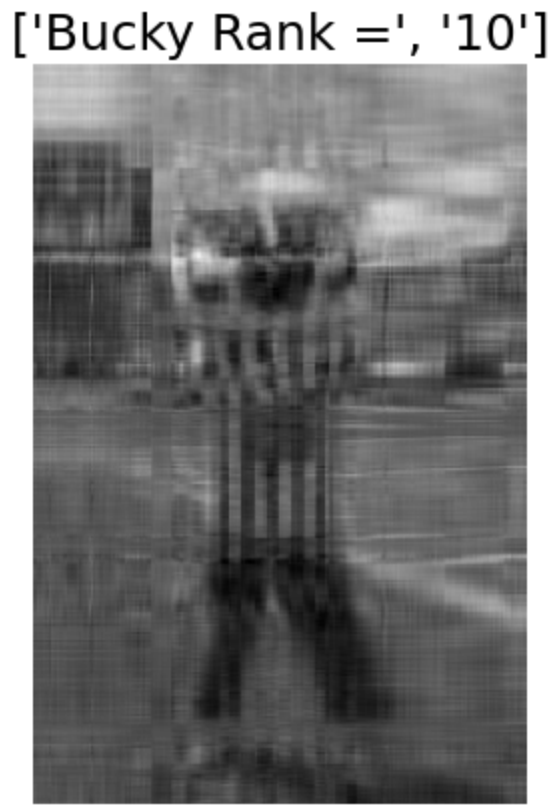
```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(np.log10(s))
ax.set_xlabel('Singular value index $i$', fontsize=16)
ax.set_ylabel('$\log_{10}(\sigma_i)$', fontsize=16)
ax.set_title('Bucky Singular Values', fontsize=18)
plt.show()
```




In [4]:
```
# Find and display low-rank approximations

r_vals = np.array([10, 20, 50, 100 ])
err_fro = np.zeros(len(r_vals))

# display images of various rank approximations
for i, r in enumerate(r_vals):

    # Complete and uncomment two lines below
    # Ar = A[:,:r]

    Ar = U[:,:r]@np.diag(s[:r])@VT[:r,:]
    print(U[:,:r].shape)
    print(np.diag(s[:r]).shape)
    print(VT[:r,:].shape)
    print(Ar.shape)
    Er = A-Ar
    err_fro[i] = np.linalg.norm(Er,ord='fro')
```

```
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.imshow(Ar,cmap='gray',interpolation='none')
    ax.set_axis_off()
    ax.set_title(['Bucky Rank =', str(r_vals[i])], fontsize=18)
    plt.show()

# plot normalized error versus rank
norm_err = err_fro/np.linalg.norm(A,ord='fro')

fig = plt.figure()
ax = fig.add_subplot(111)
ax.stem(r_vals,norm_err)
ax.set_xlabel('Rank', fontsize=16)
ax.set_ylabel('Normalized error', fontsize=16)
plt.show()
```

```
(600, 10)
(10, 10)
(10, 400)
(600, 400)
```




```
(600, 20)
(20, 20)
(20, 400)
(600, 400)
```

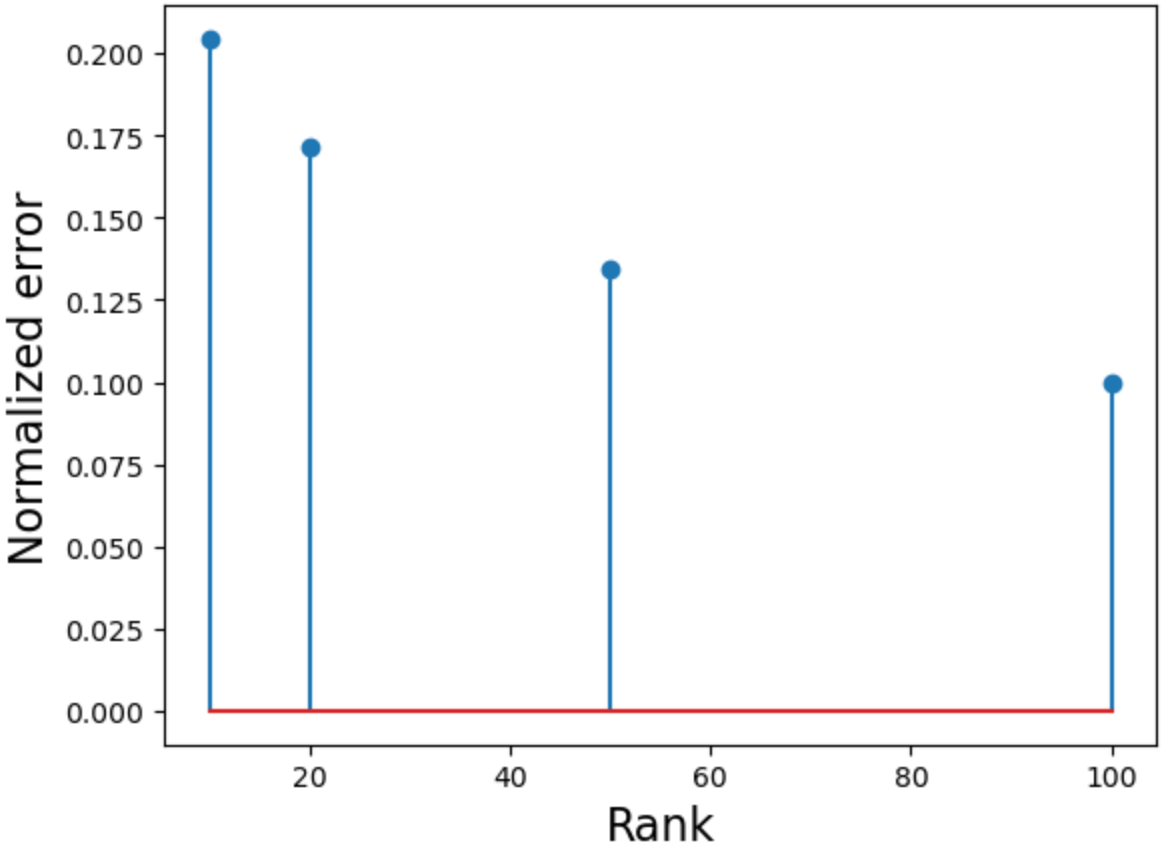
['Bucky Rank =', '20']

```
(600, 50)
(50, 50)
(50, 400)
(600, 400)
```

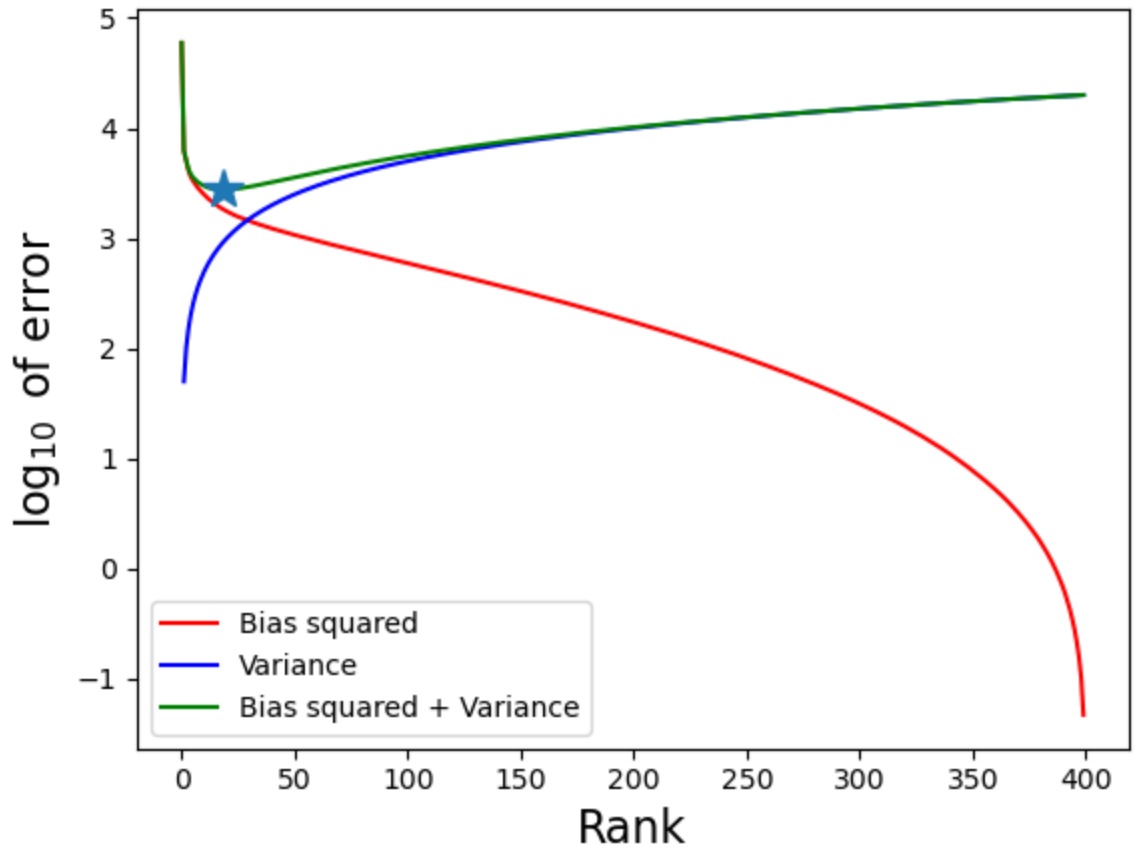
['Bucky Rank =', '50']

```
(600, 100)
(100, 100)
(100, 400)
(600, 400)
```

['Bucky Rank =', '100']



In [5]:
```
# bias-variance tradeoff
num_sv = min(rows, cols)
bias_2 = np.zeros(num_sv)
ranks = np.arange(num_sv)

for r in range(num_sv):
    bias_2[r] = np.linalg.norm(s[r:num_sv])**2

sigma2 = 50
var = sigma2*ranks
#print(var)

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(ranks,np.log10(bias_2),'r',label='Bias squared')
ax.plot(ranks[1:],np.log10(var[1:]),'b', label = 'Variance')
ax.plot(ranks,np.log10(bias_2+var),'g', label='Bias squared + Variance')
min_bias_plus_variance_index = np.argmin(np.log10(bias_2+var))
ax.plot(ranks[min_bias_plus_variance_index], np.log10(bias_2+var)[min_bias_p
ax.set_xlabel('Rank', fontsize=16)
ax.set_ylabel('$\log_{10}$ of error', fontsize=16)
ax.legend()
plt.show()
```




In [6]:
```
Anoise= A + np.sqrt(sigma2/600)*np.random.randn(np.shape(A)[0], np.shape(A)[
Anoise
```

Out[6]:
```
array([[ 1.17499756,  1.36972229,  0.93419817, ...,  0.55393274,
         0.30611841,  0.60007574],
       [ 0.73926491,  0.85861056,  1.062526  , ...,  0.70923174,
         0.33014406,  0.83035018],
       [ 1.12926001,  0.66152898,  1.1292177 , ...,  0.93548155,
         0.64496376,  0.87052681],
       ...,
       [ 0.18041518,  0.16425375,  0.33564449, ...,  0.05687059,
        -0.24820276,  1.00275999],
       [ 0.25111453, -0.09212523, -0.43009124, ..., -0.21575601,
         0.38428501,  0.35657473],
       [ 0.5349397 ,  0.36562045,  0.32224568, ...,  0.97987537,
         0.96737879, -0.21214737]])
```

In [ ]: