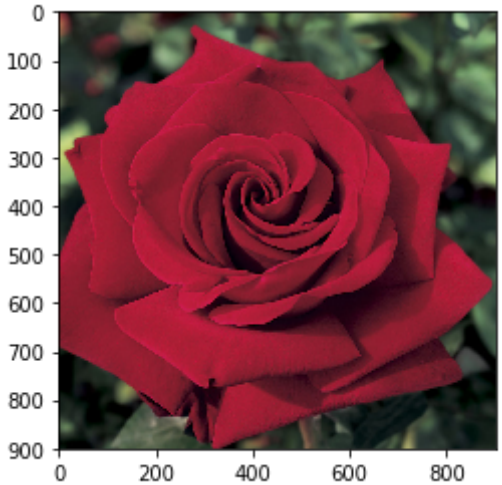
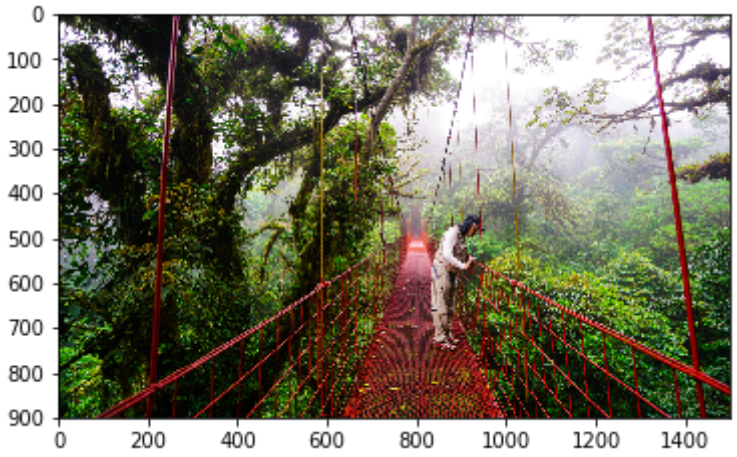
In [8]:
```
# COMP386: Computational Neuroscience
# Receptive Field HW, due 9/28/2018
#Angie Georgaras, Carolina Cervantes, Pragna Bhatt, Vaishu Pernenkil

%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import pylab as py
import math
from scipy import ndimage
```

In [9]:
```
img1=mpimg.imread('tree.jpg')
imgplot = plt.imshow(img1)
```

In [10]:
```
img2=mpimg.imread('flower.jpg')
imgplot = plt.imshow(img2)
```



In [11]:
```
img3=mpimg.imread('bridge.jpg')
imgplot = plt.imshow(img3)
```

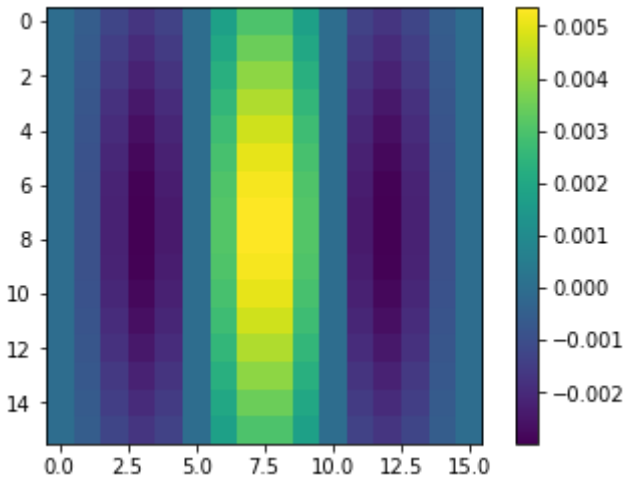


In [12]:
```
#below is the code dr.albert gave us! slightly modified with tree image
```

In [13]:
```
def gaborCalc(x, y, sx, sy, fx, fy):
    p1 = (1 / (2 * math.pi * sx * sy))
    p2 = math.exp(-0.5 * (((x * x) / (sx * sx)) + ((y * y) / (sy * sy
))))
    p3 = math.cos(2 * math.pi * (fx * x + fy * y))
    return p1 * p2 * p3

def gaborFilter(sizeX=16, sizeY=16, sx=4.0, sy=7.0, fx=0.1, fy=0.0):
    return [[gaborCalc(x - 7.5, y - 7.5, sx, sy, fx, fy) for x in range(
sizeX)] for y in range(sizeY)]
gabor_filter = gaborFilter()
print('mean before centering:', np.mean(gabor_filter))
gabor_filter = np.array(gabor_filter)
pos_ind = gabor_filter >= 0
neg_ind = gabor_filter < 0
pos_sum = np.sum(gabor_filter[pos_ind])
neg_sum = np.sum(gabor_filter[neg_ind])
gabor_filter[neg_ind] = - pos_sum / neg_sum * gabor_filter[neg_ind]
gabor_filter = list(gabor_filter)
print('mean after centering:', np.mean(gabor_filter))
py.imshow(gabor_filter)
py.colorbar()
```

```
mean before centering: 3.440428240775569e-05
mean after centering: 3.5236570605778894e-19
```

Out[13]: <matplotlib.colorbar.Colorbar at 0xb1e1aeda0>



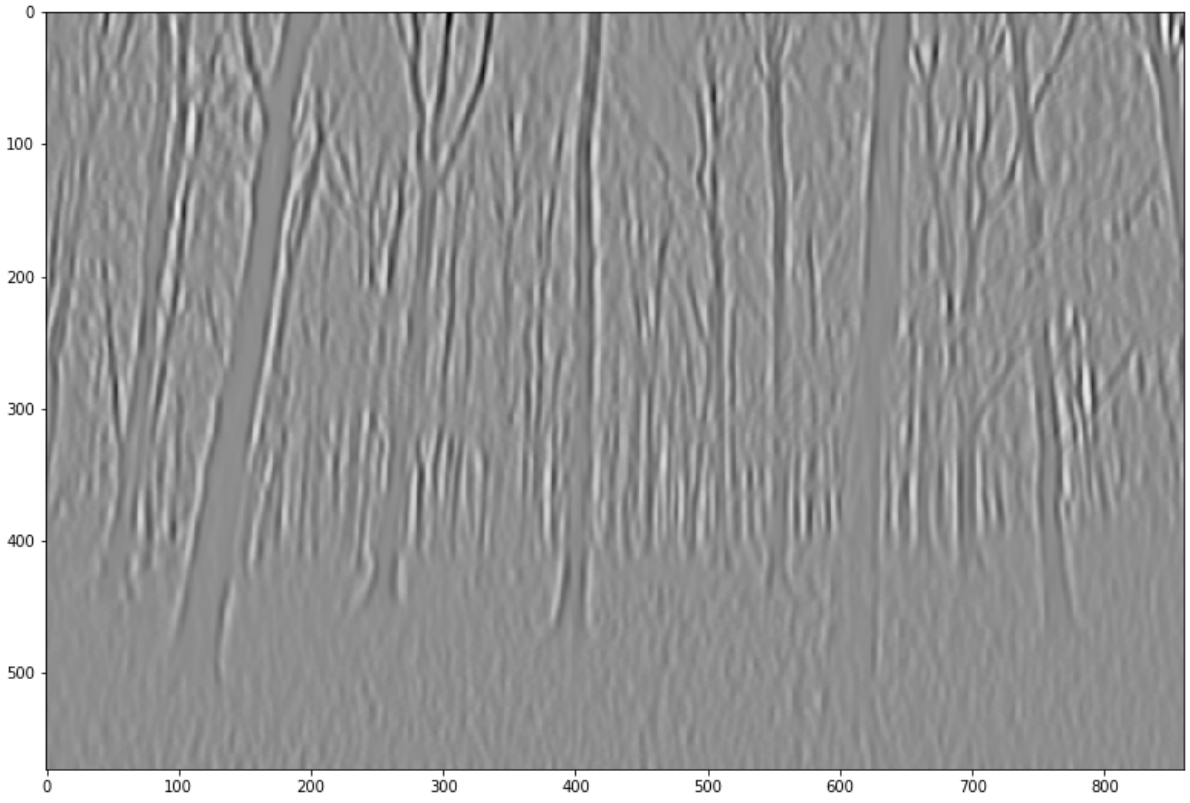
In [14]:
```
img1=mpimg.imread('tree.jpg')
img1=img1[:,:,0]
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 1)
imgplot = plt.imshow(img1, cmap=py.cm.Greys_r)
```

In [15]:
```
img2=mpimg.imread('flower.jpg')
img2=img2[:,:,0]
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 1)
imgplot = plt.imshow(img2, cmap=py.cm.Greys_r)
```

In [16]:
```
img3=mpimg.imread('bridge.jpg')
img3=img3[:,:,0]
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 1)
imgplot = plt.imshow(img3, cmap=py.cm.Greys_r)
```

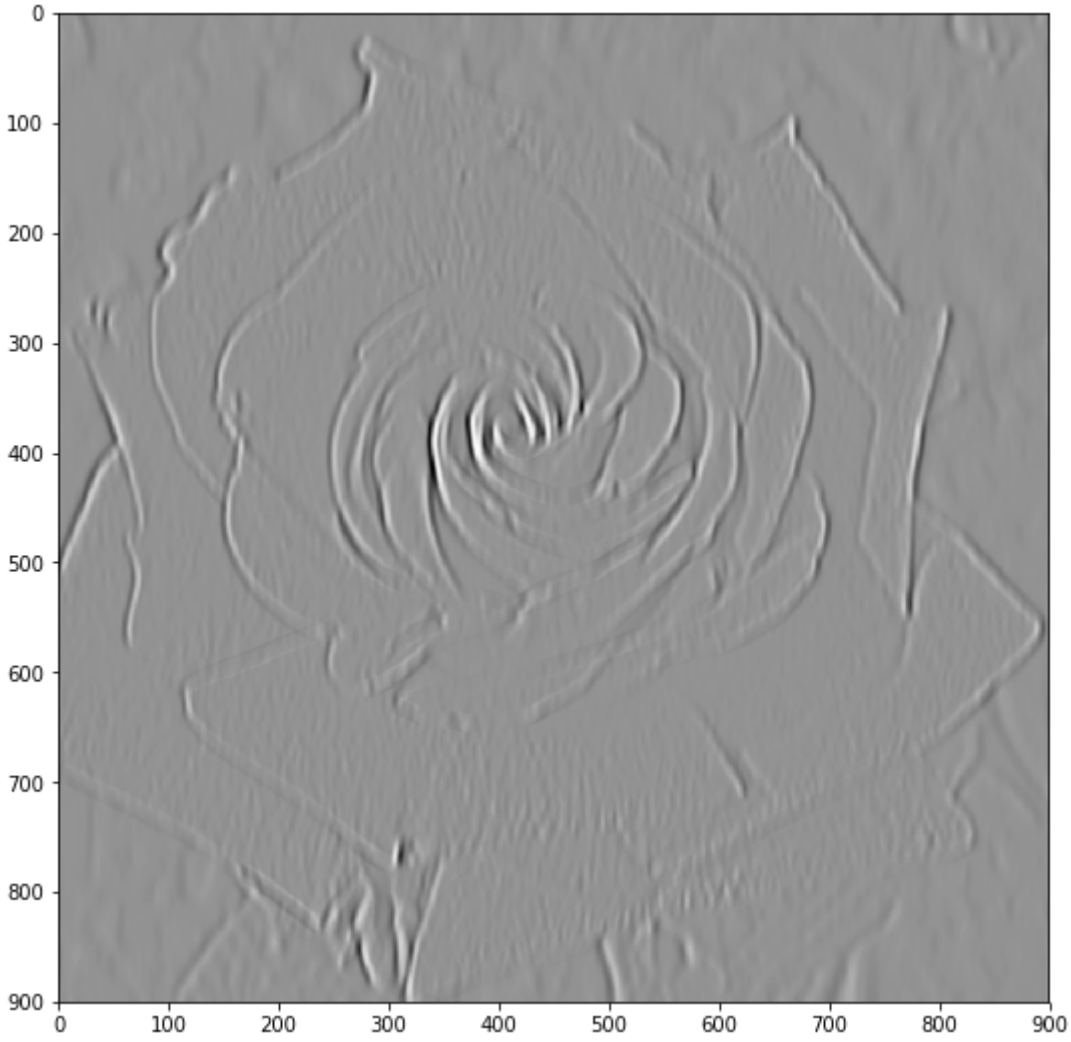
In [17]:
```
# Convolve with the gabor filter
img_gabor = ndimage.convolve(img1, gabor_filter, mode='reflect', cval=0.
0)
img_gabor = ndimage.convolve(img1.astype(float), gabor_filter, mode='ref
lect', cval=0.0)
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 2)
py.imshow(img_gabor, cmap=py.cm.Greys_r)
#Simple cell receptive field image!
```

Out[17]: <matplotlib.image.AxesImage at 0x10950bda0>



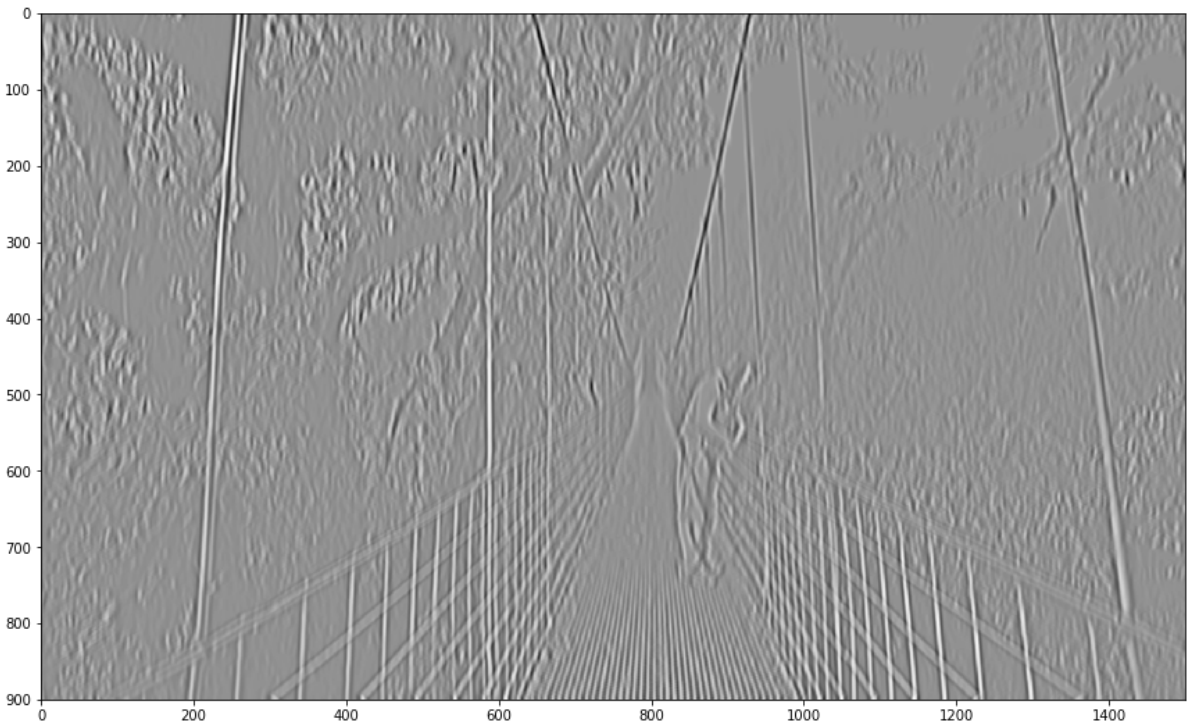
In [18]:
```
# gabor filter for image 2
img_gabor = ndimage.convolve(img2, gabor_filter, mode='reflect', cval=0.
0)
img_gabor = ndimage.convolve(img2.astype(float), gabor_filter, mode='ref
lect', cval=0.0)
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 2)
py.imshow(img_gabor, cmap=py.cm.Greys_r)
```

Out[18]: <matplotlib.image.AxesImage at 0xb1b52b080>



In [19]:
```
# gabor filter for image 3
img_gabor = ndimage.convolve(img3, gabor_filter, mode='reflect', cval=0.
0)
img_gabor = ndimage.convolve(img3.astype(float), gabor_filter, mode='ref
lect', cval=0.0)
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 2)
py.imshow(img_gabor, cmap=py.cm.Greys_r)
```

Out[19]: <matplotlib.image.AxesImage at 0xb1b5edc50>



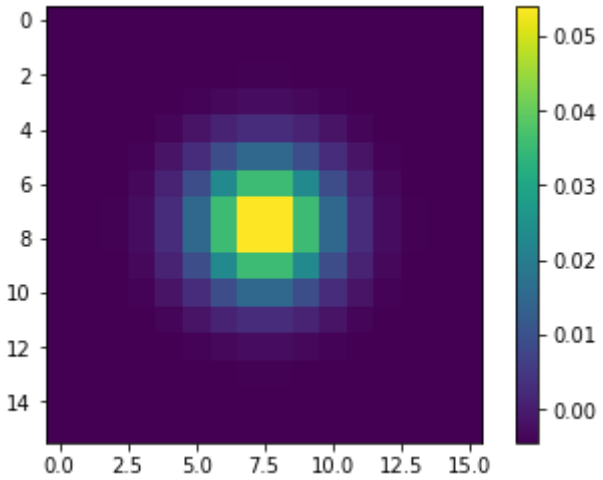
In [20]:
```
#def gaussianCalc(x, y, sigma, k):
     #return ((1 / (2 * math.pi * (math.exp(sigma))))* math.exp(-1.0 *
 ((((x * x) + (y*y))) / (math.exp (2 * sigma)))))- 1/(2 * math.pi * (k*
k) * (sigma*sigma)) * math.exp(-1.0*((x*x) + (y*y))/(2 * (k*k)*(sigma*si
gma)))
```

In [21]:
```
def gaussianCalc(x, y, sigma, k):
    return (((1/(2*math.pi*(sigma**2)))*(math.exp((-(x**2+y**2))/(2*(sig
ma**2)))))-((1/(2*math.pi*(k**2)*(sigma**2)))*math.exp((-(x**2+y**2))/(2
*(k**2)**(sigma**2)))))
```

In [22]:
```
def gaussianFilter(sizeX=16, sizeY=16, sigma = 1.6, k = 5 ):
    return [[gaussianCalc(x - 7.5, y - 7.5, sigma, k) for x in range(siz
eX)] for y in range(sizeY)]
gaussian_filter = gaussianFilter()
print('mean before centering:', np.mean(gaussian_filter))
gaussian_filter = np.array(gaussian_filter)
pos_ind = gaussian_filter >= 0
neg_ind = gaussian_filter < 0
pos_sum = np.sum(gaussian_filter[pos_ind])
neg_sum = np.sum(gaussian_filter[neg_ind])
gaussian_filter[neg_ind] = - pos_sum / neg_sum * gaussian_filter[neg_ind
]
gaussian_filter = list(gaussian_filter)
print('mean after centering:', np.mean(gaussian_filter))
py.imshow(gaussian_filter)
py.colorbar()
```

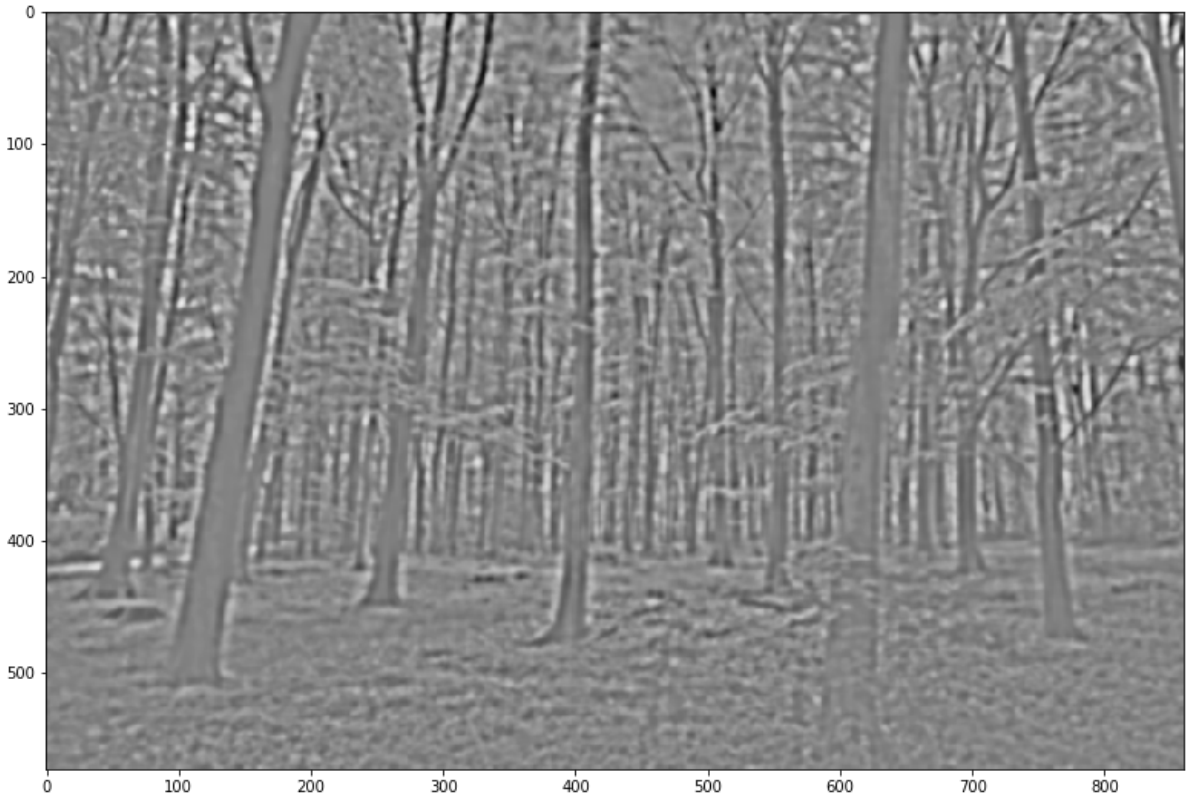
```
mean before centering: 0.001433336994448289
mean after centering: -2.168404344971009e-19
```

Out[22]: <matplotlib.colorbar.Colorbar at 0xb1af5f080>



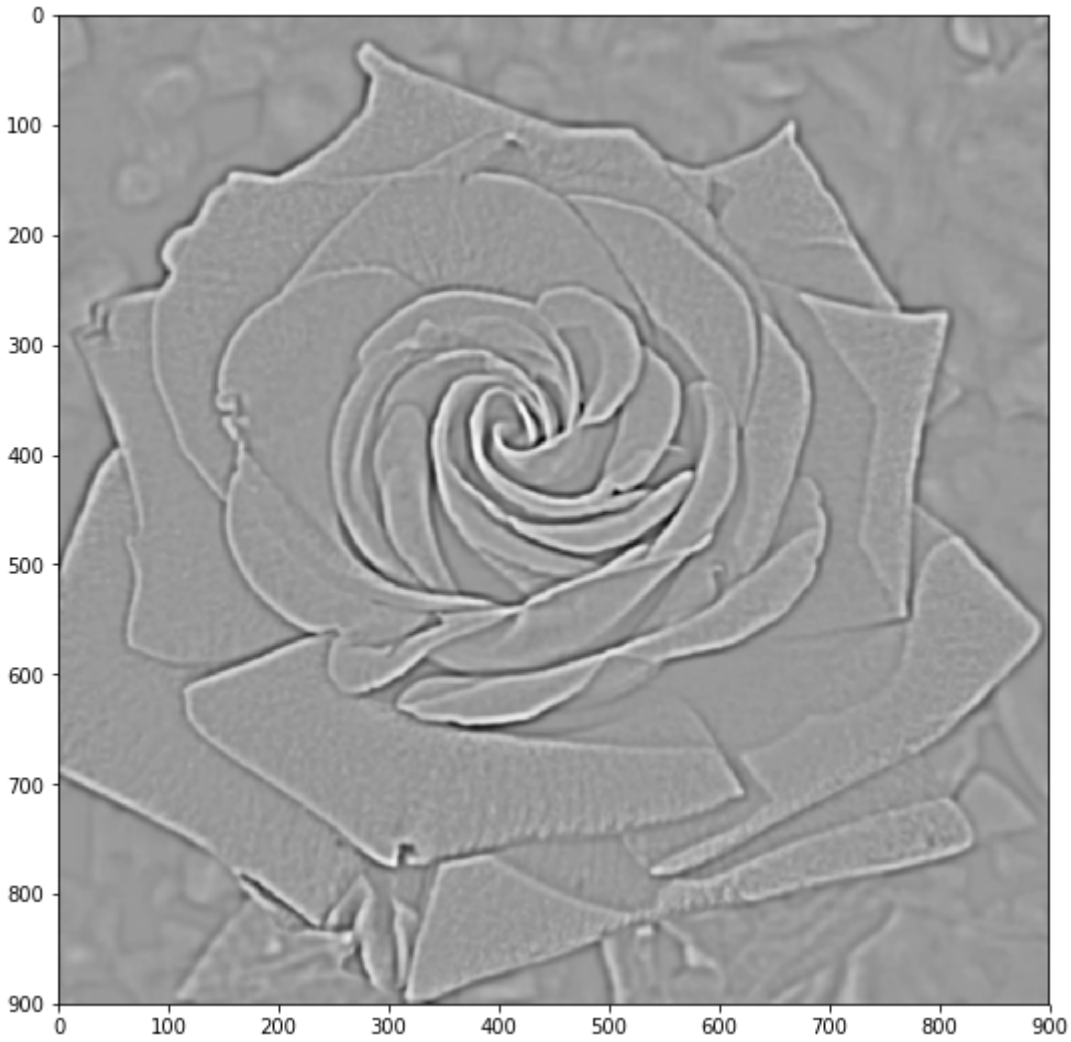
In [23]:
```
# Convolve with the gaussian filter
img_gaussian = ndimage.convolve(img1, gaussian_filter, mode='reflect', c
val=0.0)
img_gaussian = ndimage.convolve(img1.astype(float), gaussian_filter, mod
e='reflect', cval=0.0)
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 2)
py.imshow(img_gaussian, cmap=py.cm.Greys_r)
```

Out[23]: <matplotlib.image.AxesImage at 0x114002ac8>



In [24]:
```
#gaussian filter image 2
img_gaussian = ndimage.convolve(img2, gaussian_filter, mode='reflect', c
val=0.0)
img_gaussian = ndimage.convolve(img2.astype(float), gaussian_filter, mod
e='reflect', cval=0.0)
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 2)
py.imshow(img_gaussian, cmap=py.cm.Greys_r)
```

Out[24]: <matplotlib.image.AxesImage at 0x109567b38>



In [25]:
```
# gaussian filter image 3
img_gaussian = ndimage.convolve(img3, gaussian_filter, mode='reflect', c
val=0.0)
img_gaussian = ndimage.convolve(img3.astype(float), gaussian_filter, mod
e='reflect', cval=0.0)
fig = py.figure(figsize=(18, 20))
fig.add_subplot(2, 1, 2)
py.imshow(img_gaussian, cmap=py.cm.Greys_r)
```

Out[25]: <matplotlib.image.AxesImage at 0x11419acc0>