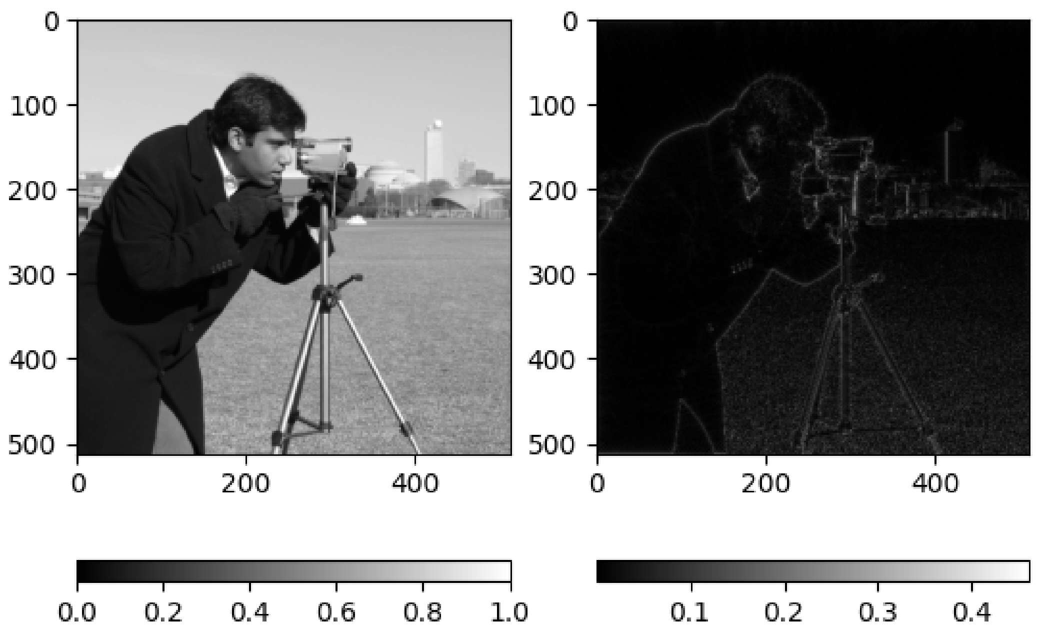
image_fourier_2

December 5, 2022

[2]:
```python
import numpy as np
import matplotlib.pyplot as plt
from skimage.draw import disk # circle - for older version of skimage
from skimage.feature import match_template
from skimage import io, data, img_as_float, img_as_ubyte, restoration
from scipy.signal import convolve2d
from scipy.fft import fft2, fftshift, ifft2
```

1 High pass filtering

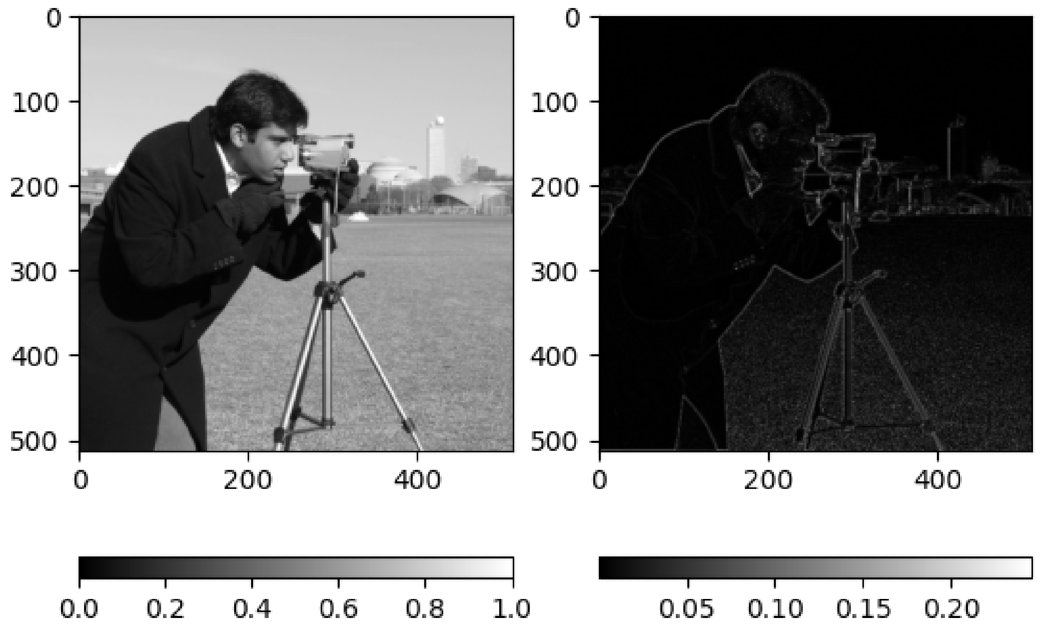
[3]:
```python
im = img_as_float(data.camera())

r = 100
imw = np.ones_like(im)
cx, cy = np.round(np.array(im.shape)/2)
# rr, cc = circle(cx, cy, r, im.shape) # for older version of skimage
rr, cc = disk((cx, cy), r, shape=im.shape)
imw[rr, cc] = 0

imfft = fftshift(fft2(im))
imm = np.log(np.abs(imfft))
imfft[imw==0] = 0
imf = np.abs(ifft2(imfft))

fig = plt.figure()
ax = fig.add_subplot(1, 2, 1)
p = plt.imshow(im, cmap='gray')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 2, 2)
p = plt.imshow(imf, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```



2 Butterworth High Pass Filter

[4]:
```
im = img_as_float(data.camera())

xs, ys = np.array(im.shape)
xl = np.linspace(0,xs,xs)
yl = np.linspace(0,ys,ys)
x, y = np.meshgrid(xl, yl)
x = (x-xs/2)/xs
y = (y-ys/2)/ys
n = 1
K = 0.1
imw = 1 / (1 + (K / (np.sqrt(x**2+y**2)**(2*n) )))

imfft = fftshift(fft2(im))
imfft = imfft * imw
imf = np.abs(ifft2(imfft))
imm = np.log(np.abs(imfft))
imm = imm * imw

fig = plt.figure()
ax = fig.add_subplot(1, 2, 1)
p = plt.imshow(im, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```

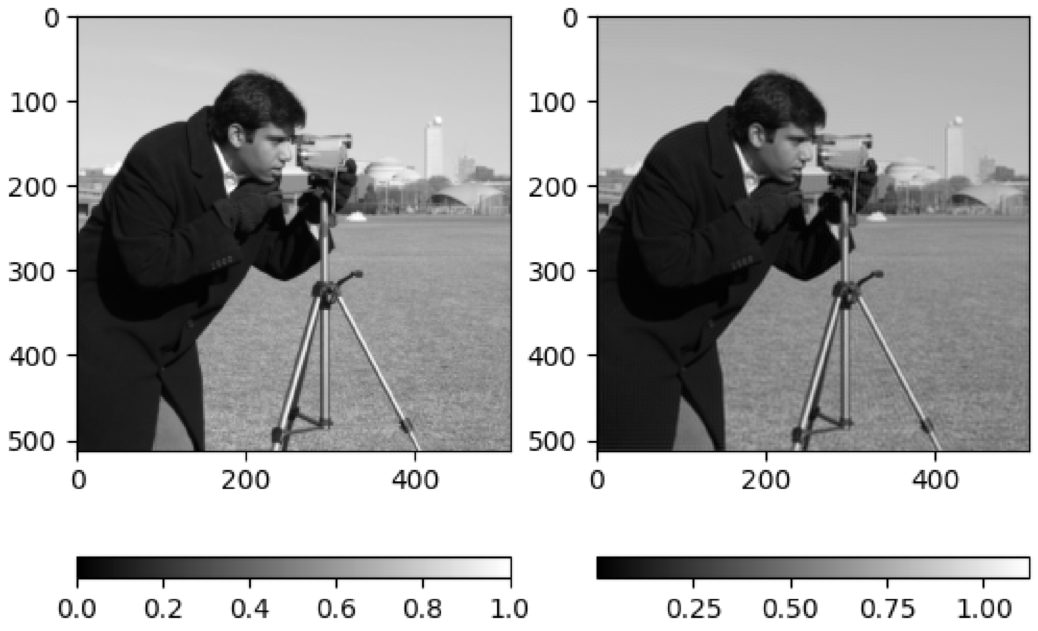
```
ax = fig.add_subplot(1, 2, 2)
p = plt.imshow(imf, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```



3 Low pass filtering

[6]:
```
im = img_as_float(data.camera())

r = 100
imw = np.zeros_like(im)
cx, cy = np.round(np.array(im.shape)/2)
# rr, cc = circle(cx, cy, r, im.shape) # for older version of skimage
rr, cc = disk((cx, cy), r, shape=im.shape)
imw[rr, cc] = 1

imfft = fftshift(fft2(im))
imm = np.log(np.abs(imfft))
imfft[imw==0] = 0
imf = np.abs(ifft2(imfft))

fig = plt.figure()
ax = fig.add_subplot(1, 2, 1)
p = plt.imshow(im, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```

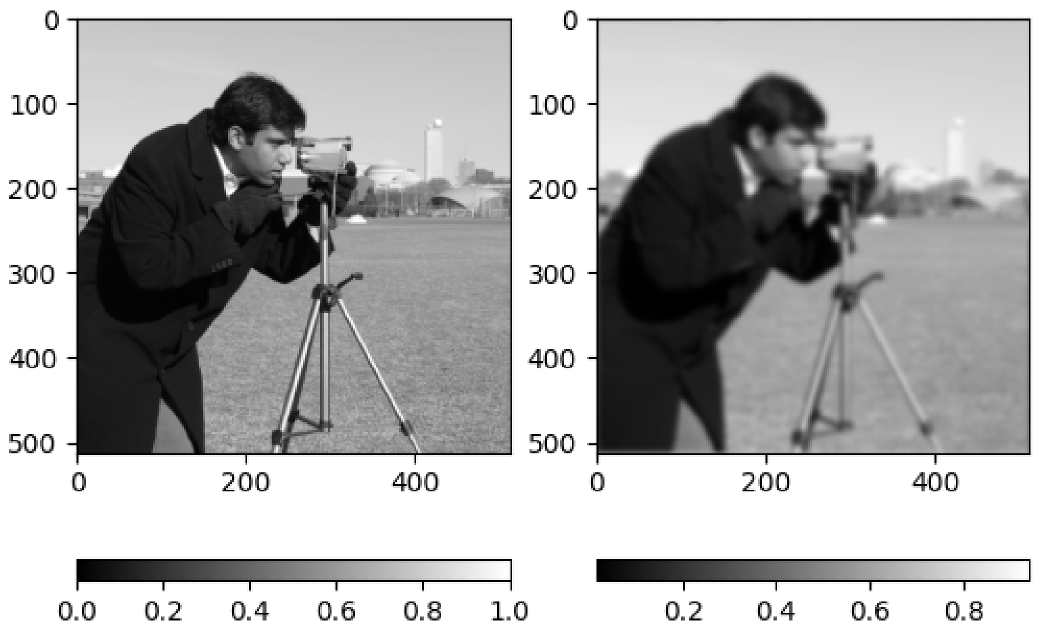
```
ax = fig.add_subplot(1, 2, 2)
p = plt.imshow(imf, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```



4 Gaussian Low Pass Filter

[7]:
```
im = img_as_float(data.camera())

xs, ys = np.array(im.shape)
xl = np.linspace(0,xs,xs)
yl = np.linspace(0,ys,ys)
x, y = np.meshgrid(xl, yl)
x = (x-xs/2)/xs
y = (y-ys/2)/ys
s = 0.05
imw = np.exp(-(x**2+y**2)/(2*s**2))
# imw = np.exp(-((x/s)**2+(y/s)**2))

imfft = fftshift(fft2(im))
imfft = imfft * imw
imf = np.abs(ifft2(imfft))
imm = np.log(np.abs(imfft))
imm = imm * imw

fig = plt.figure()
```

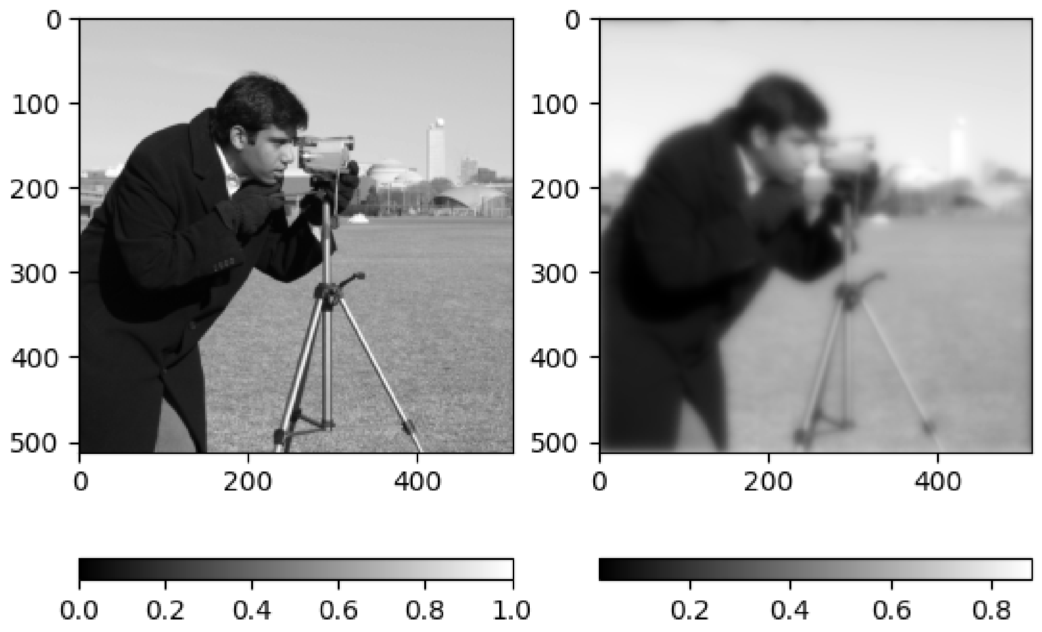
```python
ax = fig.add_subplot(1, 2, 1)
p = plt.imshow(im, cmap='gray')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 2, 2)
p = plt.imshow(imf, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```



5 Butterworth Low Pass Filter

[8]:
```python
im = img_as_float(data.camera())

xs, ys = np.array(im.shape)
xl = np.linspace(0,xs,xs)
yl = np.linspace(0,ys,ys)
x, y = np.meshgrid(xl, yl)
x = (x-xs/2)/xs
y = (y-ys/2)/ys
n = 1
K = 0.001
imw = 1 / (1 + ((np.sqrt(x**2+y**2)**(2*n) / K)))

imfft = fftshift(fft2(im))
imfft = imfft * imw
imf = np.abs(ifft2(imfft))
```

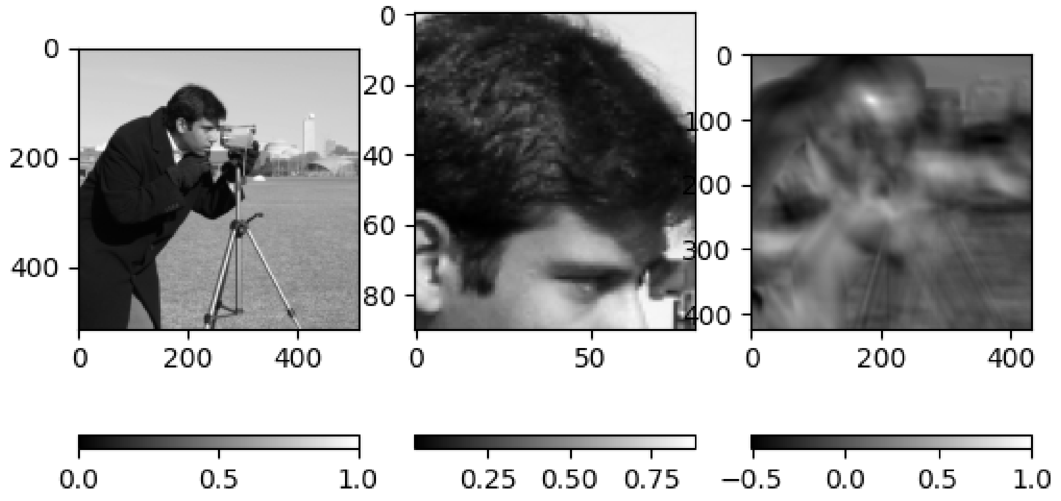
```
imm = np.log(np.abs(imfft))
imm = imm * imw

fig = plt.figure()
ax = fig.add_subplot(1, 2, 1)
p = plt.imshow(im, cmap='gray')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 2, 2)
p = plt.imshow(imf, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```



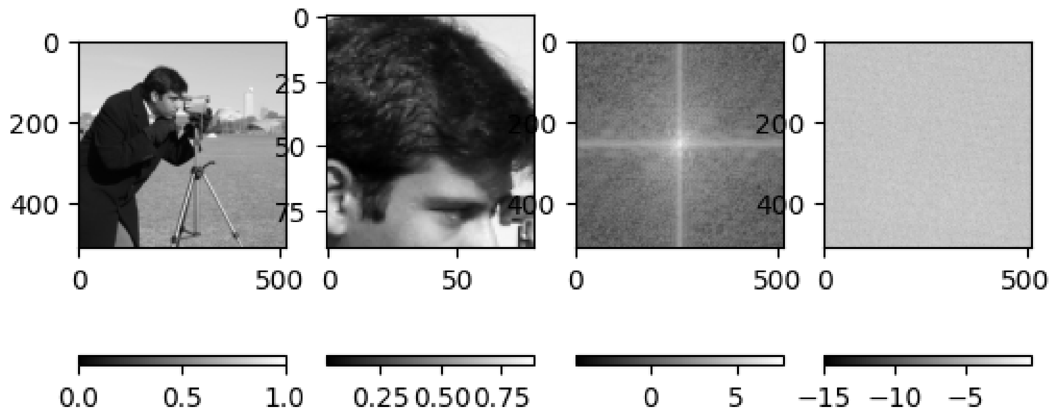
6 Correlation

[9]:
```
im = img_as_float(data.camera())
imtemp = im[70:160,186:266]
imm = match_template(im, imtemp)

fig = plt.figure()
ax = fig.add_subplot(1, 3, 1)
p = plt.imshow(im, cmap='gray')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 3, 2)
p = plt.imshow(imtemp, cmap='gray')
```

```
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 3, 3)
p = plt.imshow(imm, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```

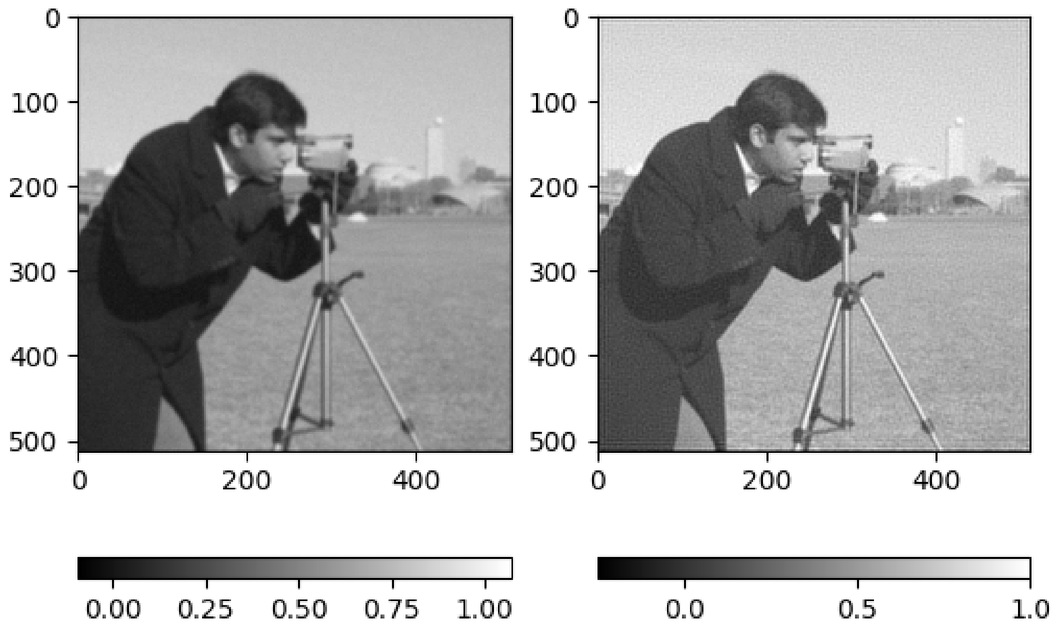


[10]:
```
im = img_as_float(data.camera())
imtemp = im[70:160,186:266]

imm = np.log(np.abs(ifft2(np.divide(fft2(im), fft2(imtemp,im.shape)))))
imtempf = np.log(np.abs(fftshift(fft2(imtemp,im.shape))))

fig = plt.figure()
ax = fig.add_subplot(1, 4, 1)
p = plt.imshow(im, cmap='gray')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 4, 2)
p = plt.imshow(imtemp, cmap='gray')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 4, 3)
p = plt.imshow(imtempf, cmap='gray')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 4, 4)
p = plt.imshow(imm, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```



7 Deconvolution

[11]:
```
im = img_as_float(data.camera())
kernel = np.ones((5,5)) / 25
im = convolve2d(im, kernel, 'same')

im += 0.1 * im.std() * np.random.standard_normal(im.shape)
imf, _ = restoration.unsupervised_wiener(im, kernel)

fig = plt.figure()
ax = fig.add_subplot(1, 2, 1)
p = plt.imshow(im, cmap='gray')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 2, 2)
p = plt.imshow(imf, cmap='gray')
c = plt.colorbar(orientation='horizontal')
```


0
100
200
300
400
500
0
200
400
0.00
0.25
0.50
0.75
1.00
0
100
200
300
400
500
0
200
400
0.0
0.5
1.0

```
[ ]:
```