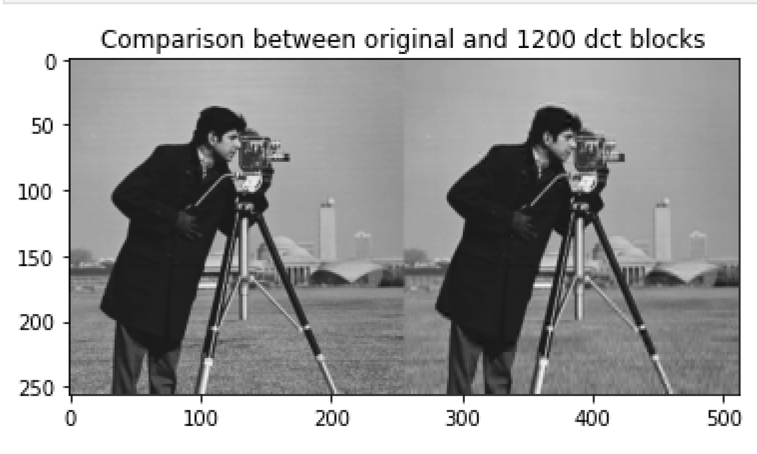
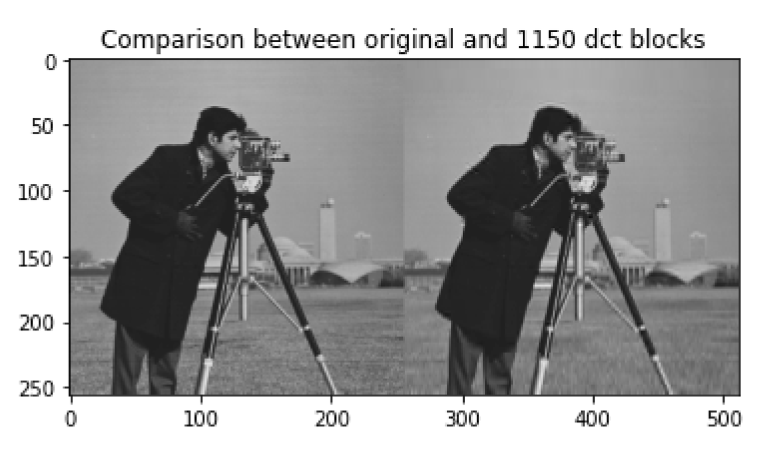
In [227…

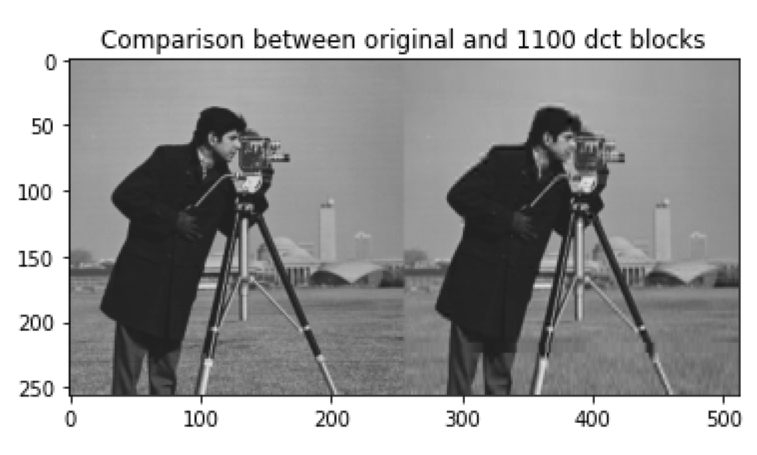
```python
from random import gauss

import cv2
from matplotlib.pyplot import figure, subplot, title, axis, imshow, show, hist
from numpy import uint8, zeros, r_, max, argpartition, column_stack, unravel_index, arange, hstack
from scipy.fftpack import dct, idct

# Hiding secret message in DCT

figure_size = (18, 7)
def dct2(a):
    return dct(dct(a.T, norm='ortho').T, norm='ortho')
def idct2(a):
    return idct(idct(a.T, norm='ortho').T, norm='ortho')
def show_images(name, images, desc, col=None):
    figure(figsize=figure_size)
    n = len(images)
    for k in range(0, n):
        img = images[k].astype(uint8)
        tit = name + " " + desc[k]
        if col is not None:
            c = col[k]
            subplot(1, n, k + 1), title(tit), axis('off'), imshow(img, cmap=c)
        else:
            subplot(1, n, k + 1), title(tit), axis('off'), imshow(img)
    show()
def find_dct_blocks(img):
    dct_blocks = zeros(img.shape)
    for i in r_[:img.shape[0]:8]:
        for j in r_[:img.shape[1]:8]:
            dct_blocks[i:i + 8, j:j + 8] =dct2(img[i:i + 8,j:j + 8])

    return dct_blocks
def k_largest_values(a, K):
    idx = argpartition(a.ravel(), a.size-K)[-K:]
    return column_stack(unravel_index(idx, a.shape))
def find_best_k_dct(_dct, img):
    size = img.shape
    _k_dct = None
    _k_idx = None
    _k = -1
    for K in range(1200,1000,-50):
        k_idct = zeros(sz)
        idx = k_largest_values(_dct, K)
        filtered = zeros(_dct.shape)
        filtered[idx] = _dct[idx]

        for i in r_[:size[0]:8]:
            for j in r_[:size[1]:8]:
                k_idct[i:(i+8),j:(j+8)] = idct2(filtered[i:(i+8),j:(j+8)] )

        figure()
        imshow(hstack( (img, k_idct) ) ,cmap='gray')
        title(str("Comparison between original and %d dct blocks" %K ))

        if K == 1100 :  # best one found by inspection
            _k = K
            _k_dct = k_idct
            _k_idx = idx

    return _k, _k_dct, _k_idx
```

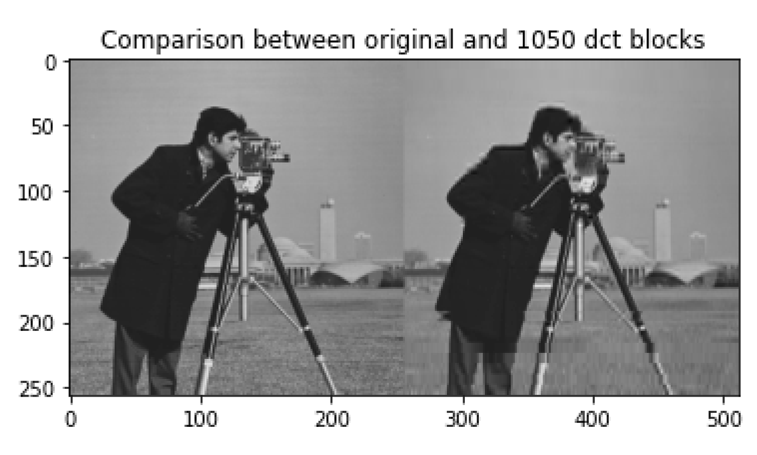
In [228…

```python
image = cv2.imread("images/cameraman.tif", 0)
sz = image.shape
_dct = find_dct_blocks(image)
#test
k, k_dct, k_idx = find_best_k_dct(_dct, image)
```







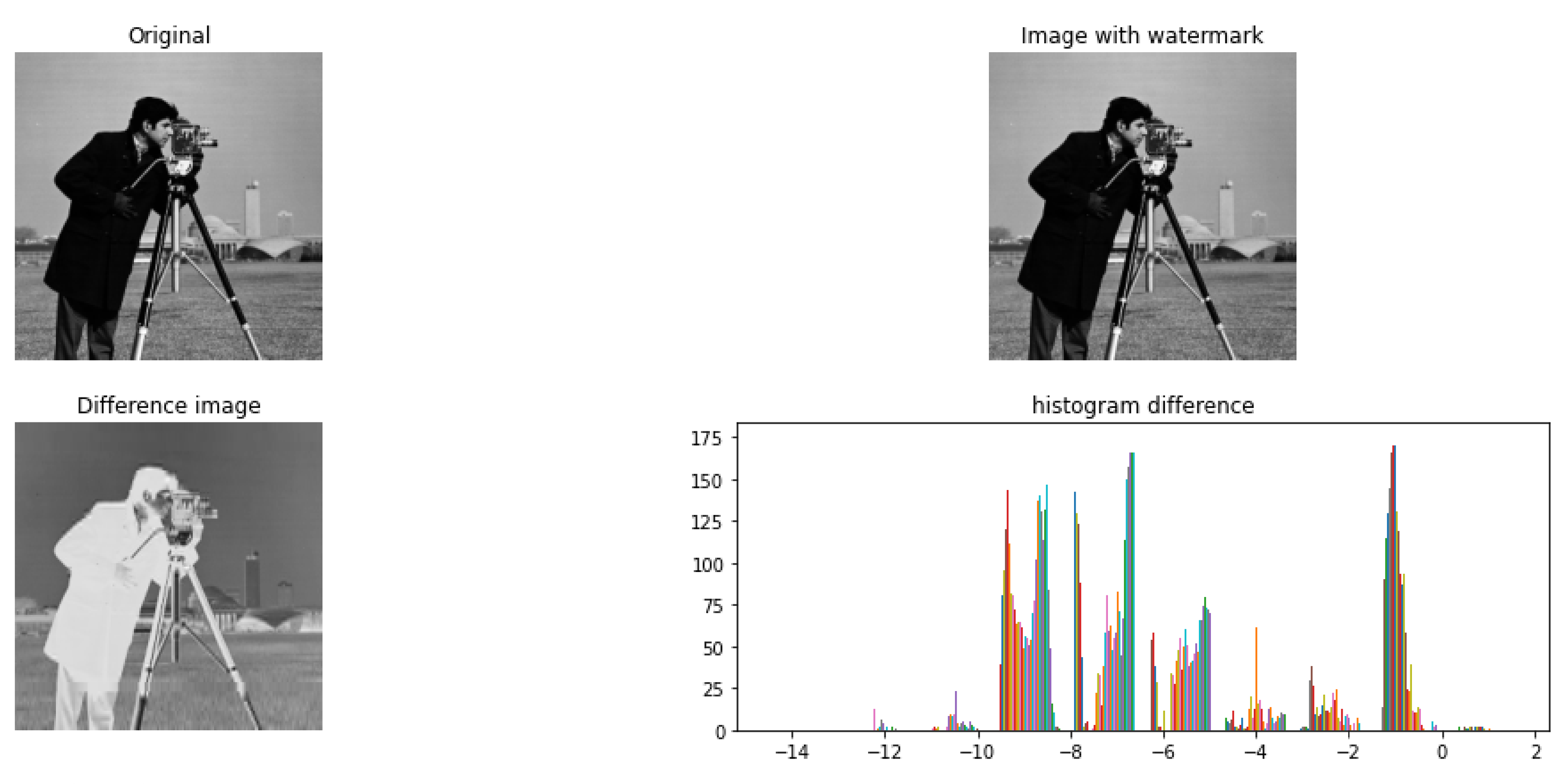


insert watermark

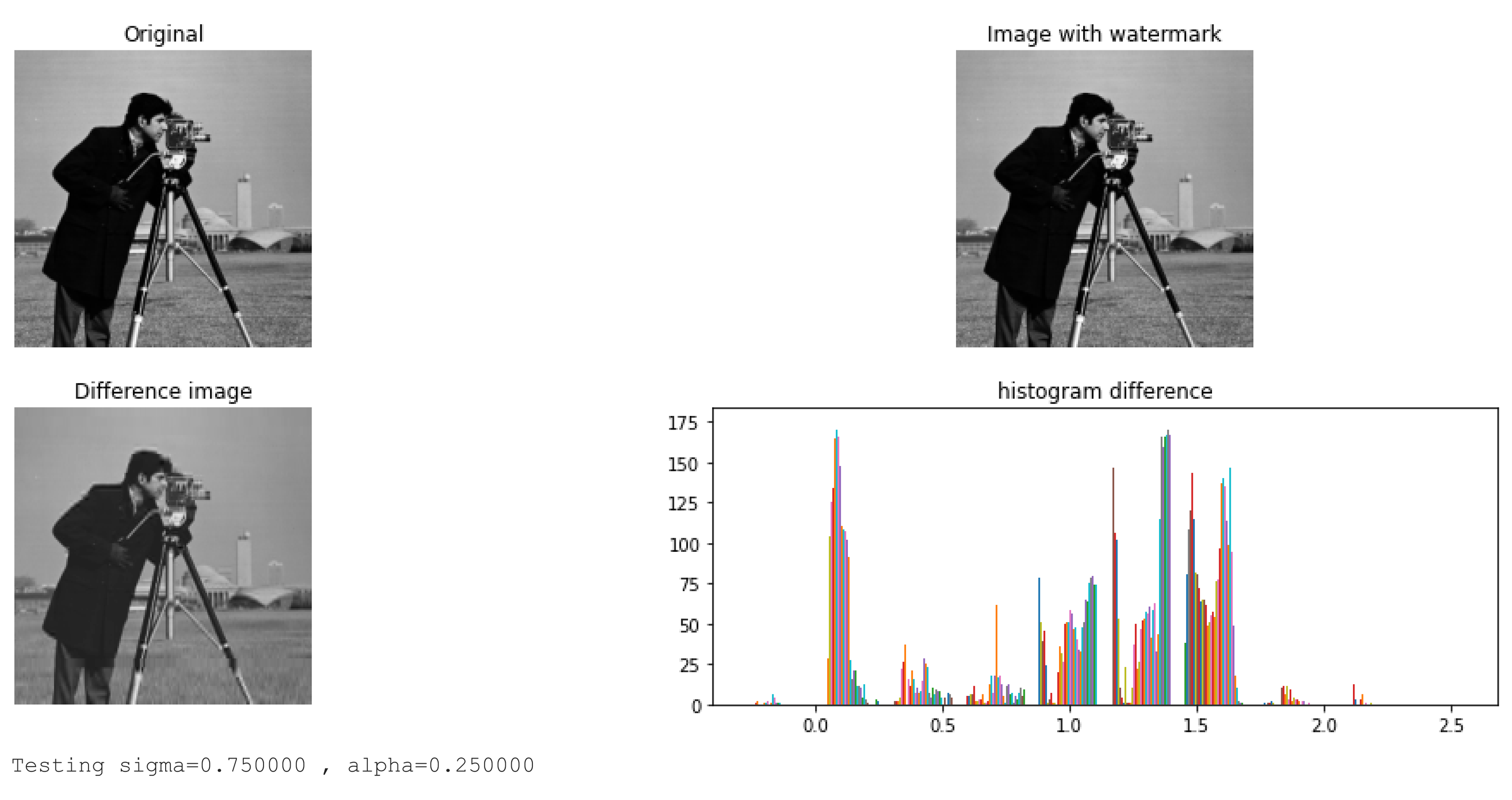
In [229…

```python
def insert_watermark(img, c, idx, s, a):
    wk = zeros(k_dct.shape)
    wk[idx] = gauss(mu=0,sigma=s)  # zeros everywhere but in the positions of the k highest values
    # hist(wk , bins=100), show()
    marked = c * (1 + a * wk)
    i_marked = zeros(sz)
    for i in r_[:sz[0]:8]:
        for j in r_[:sz[1]:8]:
            i_marked[i:(i+8),j:(j+8)] = idct2(marked[i:(i+8),j:(j+8)])

    diff = img-i_marked
    print("\nTesting sigma=%f , alpha=%f \n" %(s,a))
    figure(figsize=figure_size)
    subplot(2,2,1), axis("off"), title("Original") , imshow(img, cmap='gray')
    subplot(2,2,2), axis("off"), title("Image with watermark") , imshow(i_marked, cmap='gray')
    subplot(2,2,3), axis("off"), title("Difference image") , imshow(diff, cmap='gray')
    subplot(2,2,4), title("histogram difference") , hist(diff)
    show()
    return i_marked

#test
for alpha in [0.25, 0.5, 0.75]:
    for sigma in [0.25, 0.5, 0.75]:
        _idct = insert_watermark(image, _dct, k_idx, sigma, alpha)

#looks invisible to me anyway
# keeping watermarked image with sigma = 0.05, alpha = 0.975
sigma = 2
alpha = 0.975
```

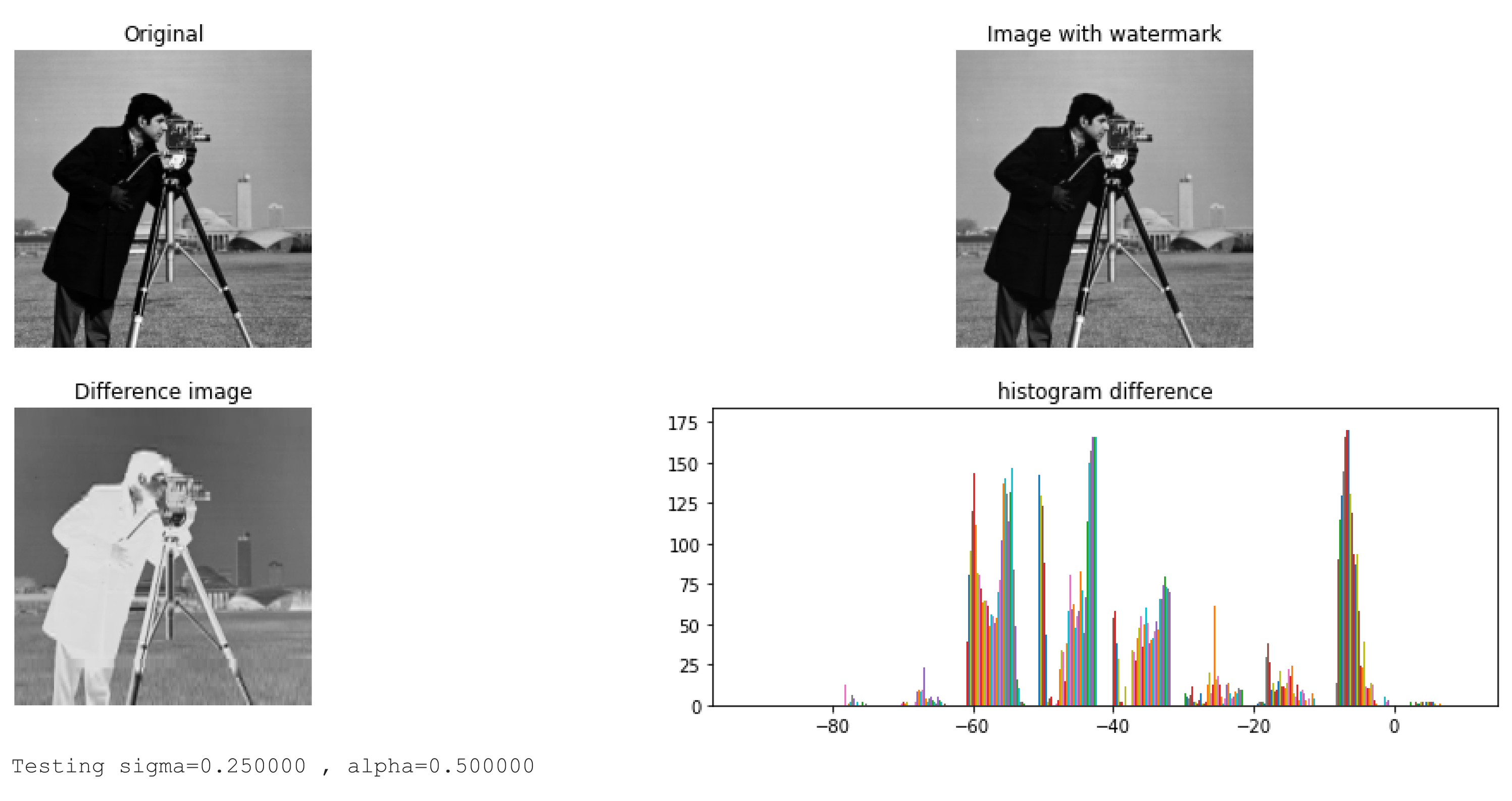
```
Testing sigma=0.250000 , alpha=0.250000
```



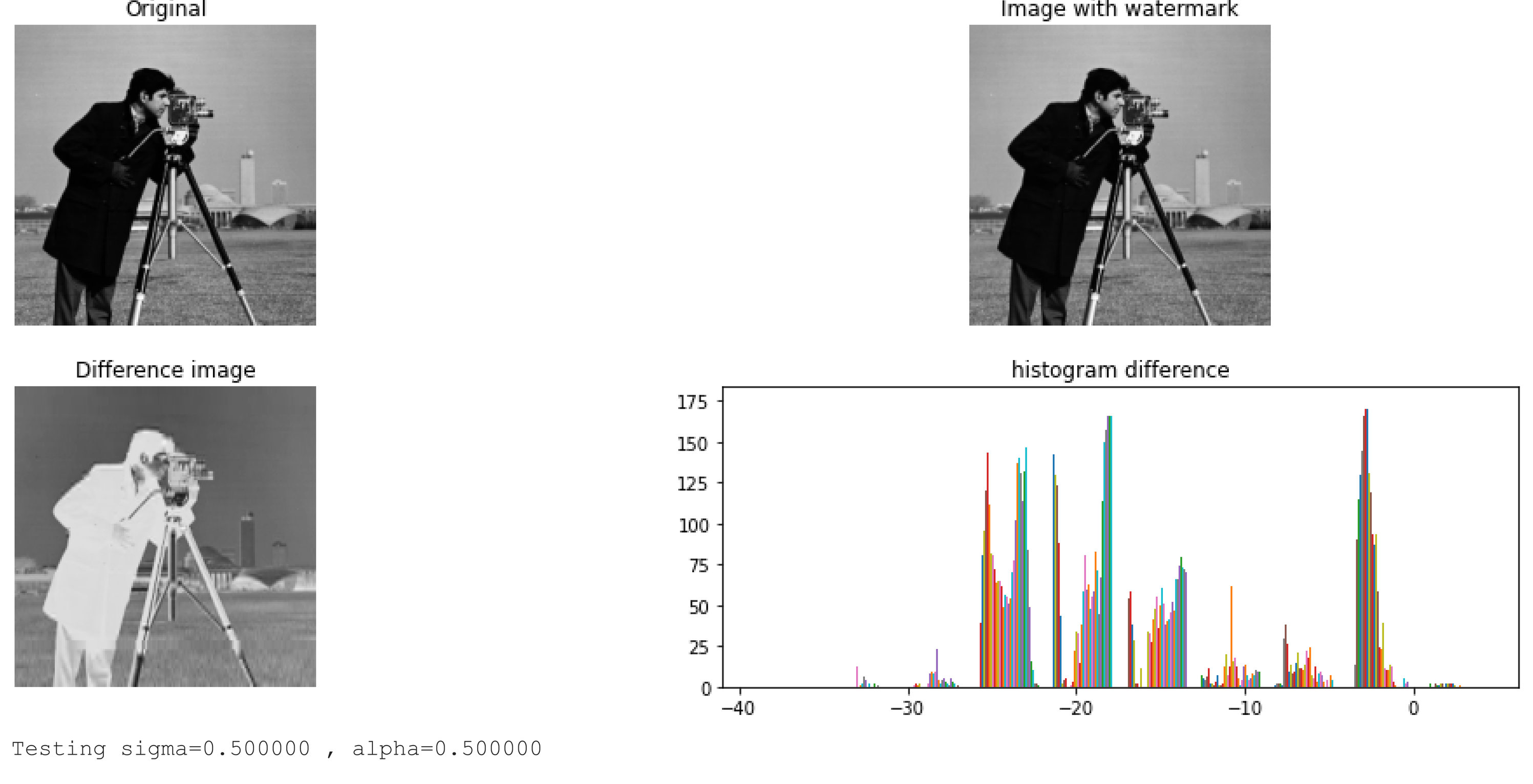

```
Testing sigma=0.500000 , alpha=0.250000
```



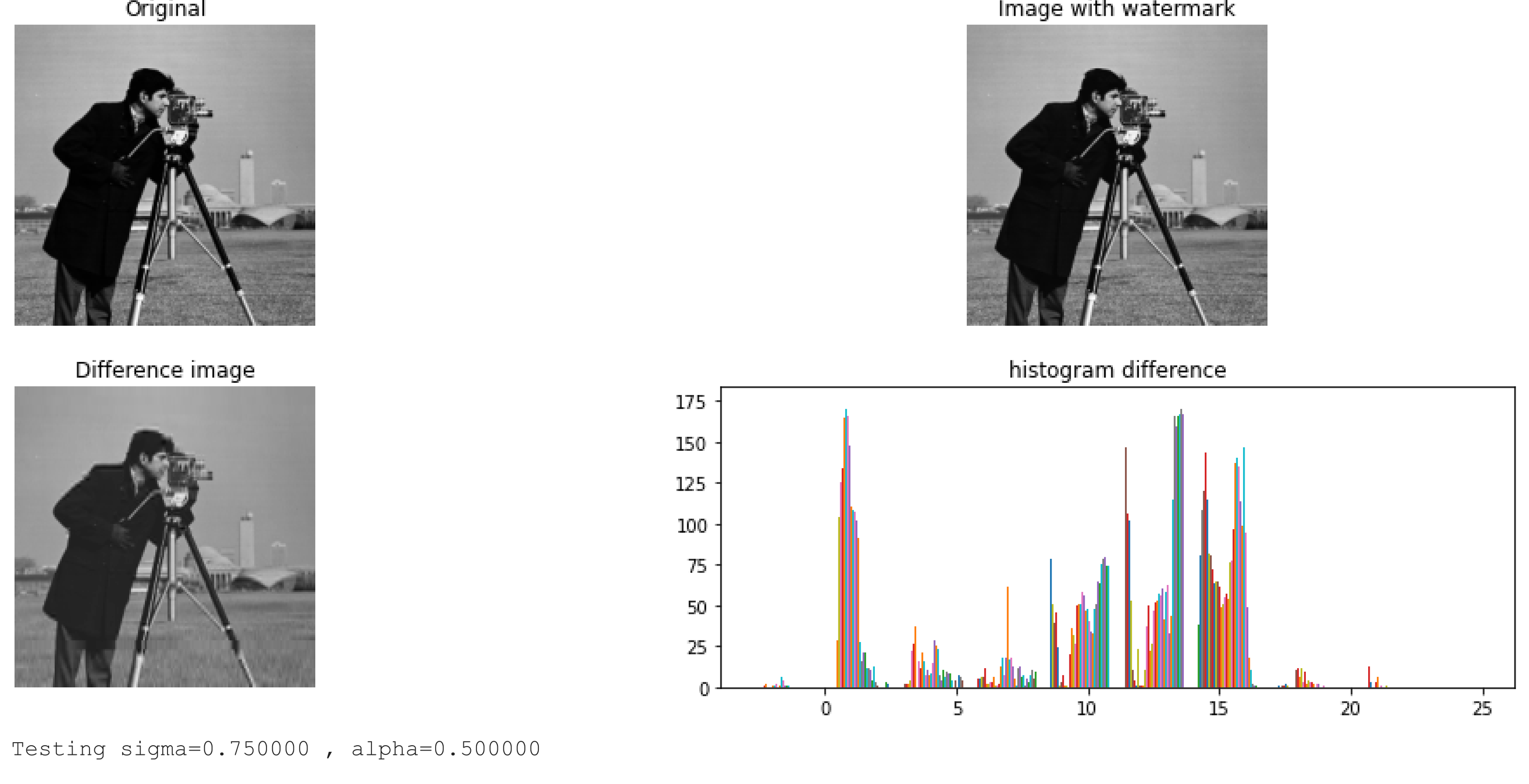

```
Testing sigma=0.750000 , alpha=0.250000
```



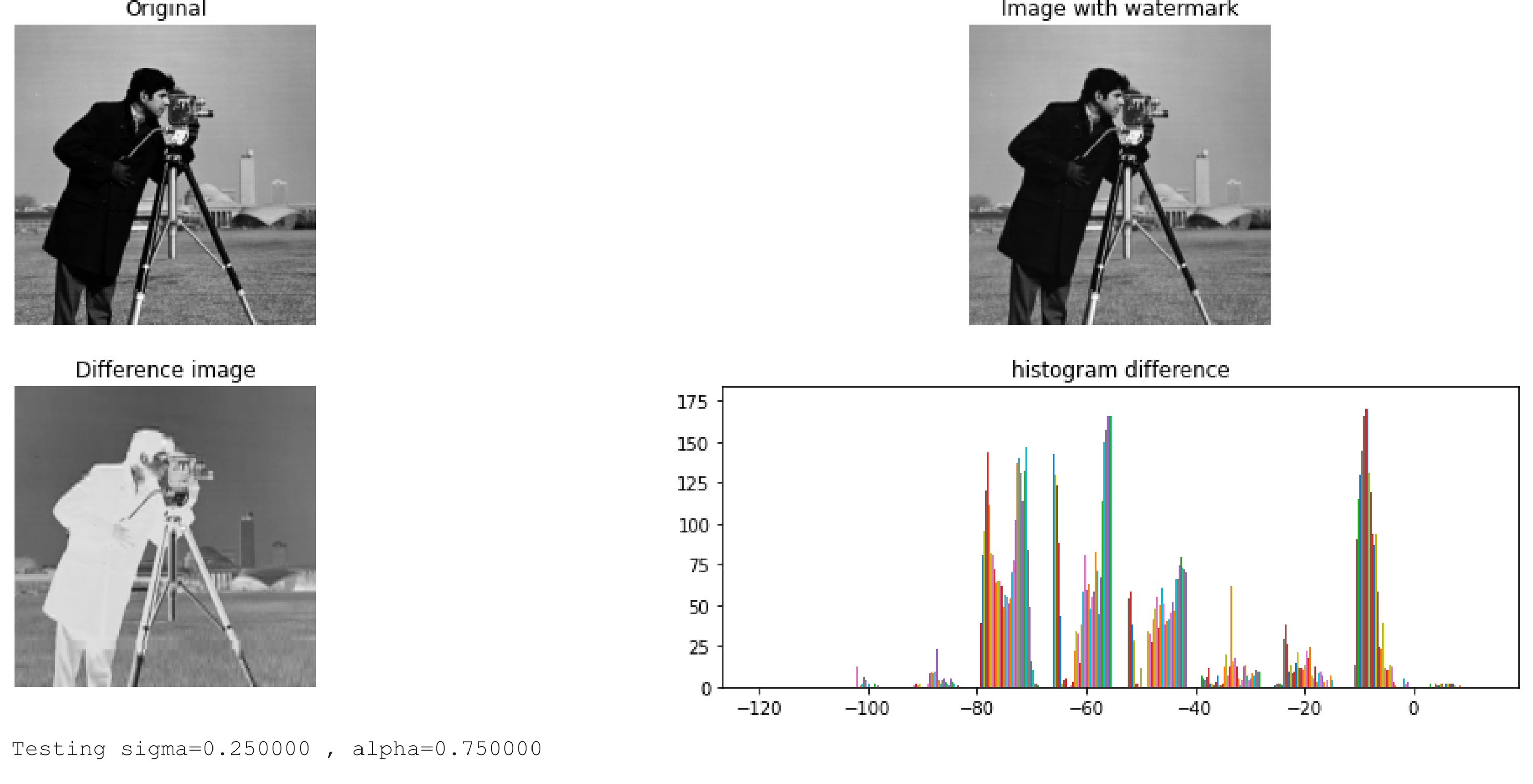

```
Testing sigma=0.250000 , alpha=0.500000
```



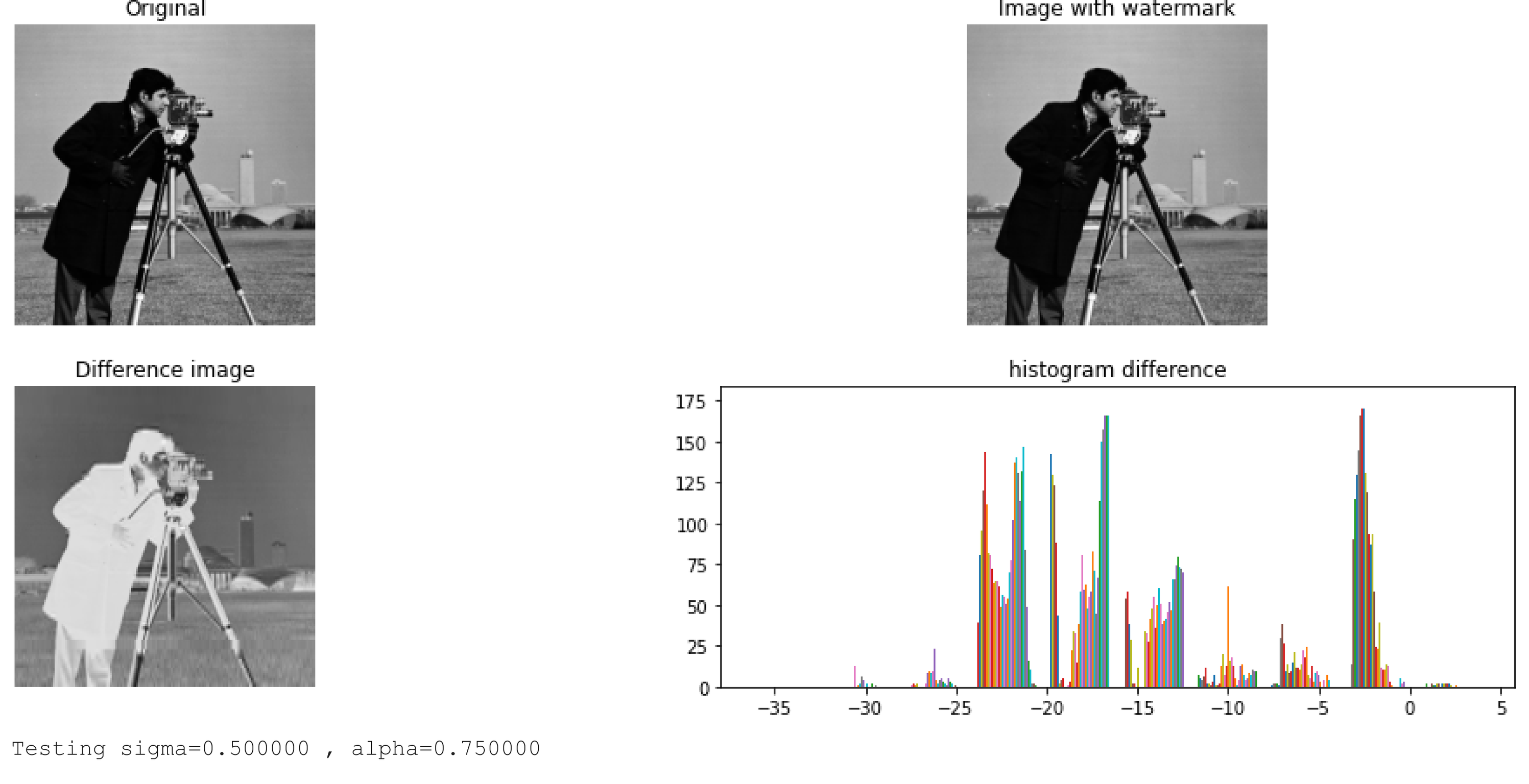

```
Testing sigma=0.500000 , alpha=0.500000
```



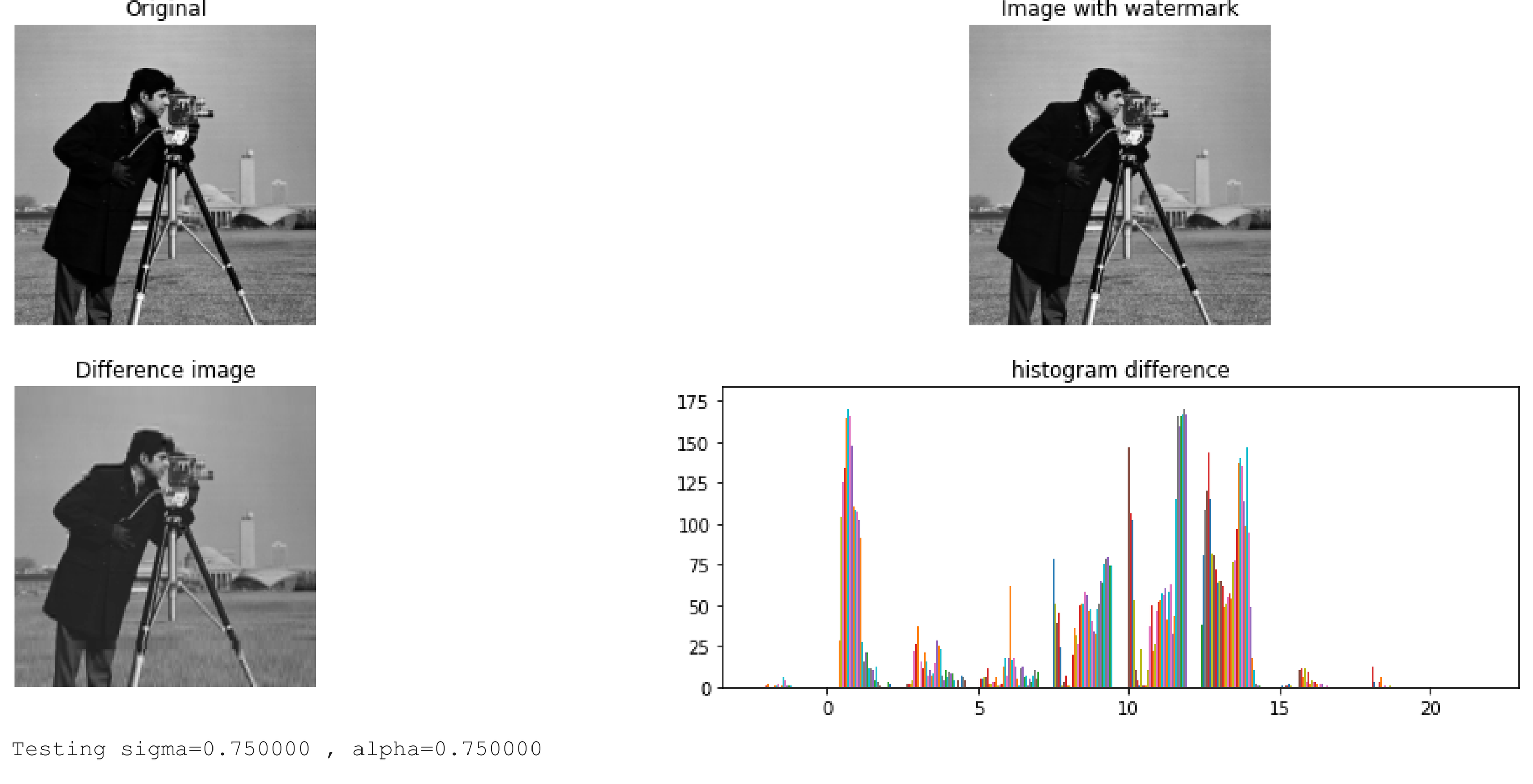

```
Testing sigma=0.750000 , alpha=0.500000
```



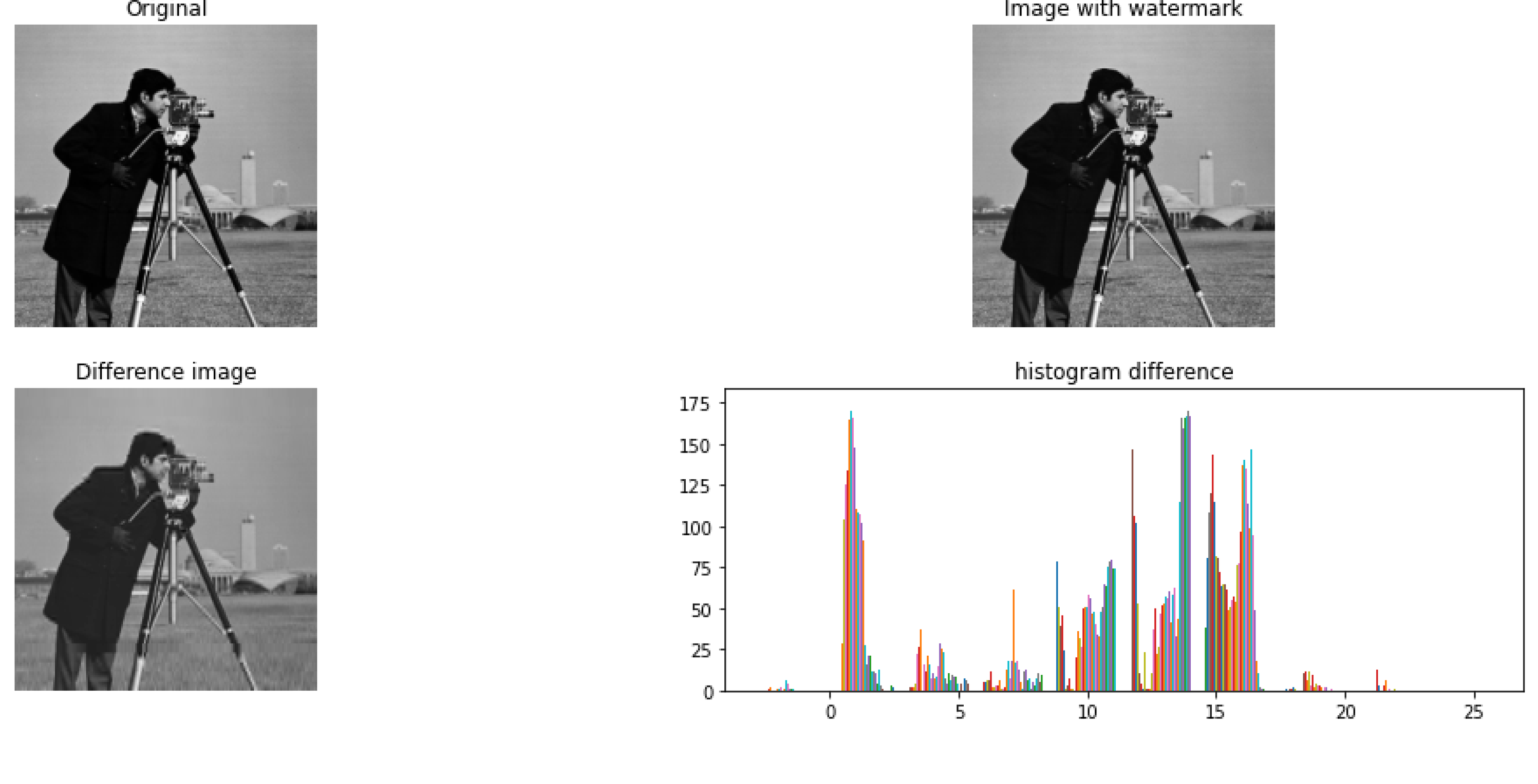

```
Testing sigma=0.250000 , alpha=0.750000
```




```
Testing sigma=0.500000 , alpha=0.750000
```




```
Testing sigma=0.750000 , alpha=0.750000
```



Part 2

CASE 1

Mistery image is the watermarked image

In [230…

```python
mistery = _idct
dct_m = find_dct_blocks(mistery)
```

CASE 2

Mistery image is the original image