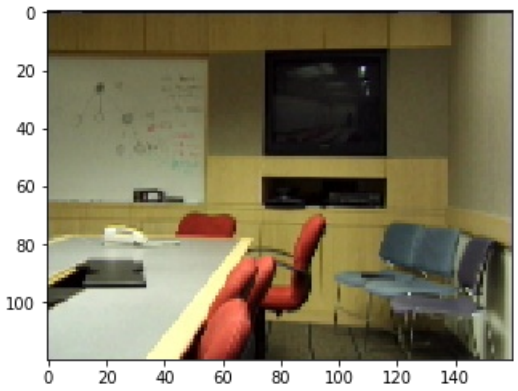
In [1]:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

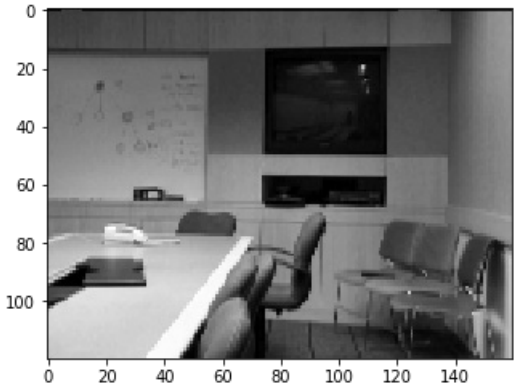
In [2]:

```
image = cv2.imread("D:\\computer vision\\MovedObject\\b00001.bmp")
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.show()
```



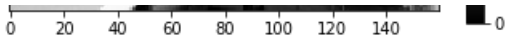
In [3]:

```
image = cv2.imread("D:\\computer vision\\MovedObject\\b00001.bmp", 0)
plt.imshow(image, cmap='gray')
plt.show()
```



In [4]:

```
image2 = cv2.imread("D:\\computer vision\\MovedObject\\b00660.bmp", 0)
plt.imshow(image2, cmap="gray")
plt.colorbar()
plt.show()
```

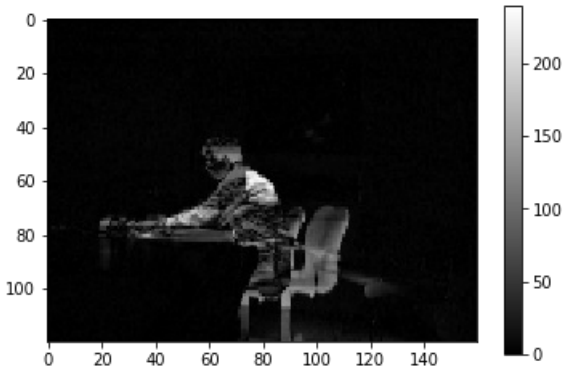


In [5]:

```
image3 = cv2.imread("D:\\computer vision\\MovedObject\\b00001.bmp", 0)
N = image.shape[0]*image.shape[1]
k = N

for i in range(image.shape[0]):
    for j in range(image.shape[1]):

        if image[i][j] < image2[i][j]:
            image3[i][j] = image2[i][j] - image[i][j]
        else:
            image3[i][j] = image[i][j] - image2[i][j]

#          if(image3[i][j] < 20):
#              image3[i][j] = 0
#              k= k-1
```

In [7]:

```
#image3 = image2 - image
plt.imshow(image3, cmap="gray")
plt.colorbar()
plt.show()
```

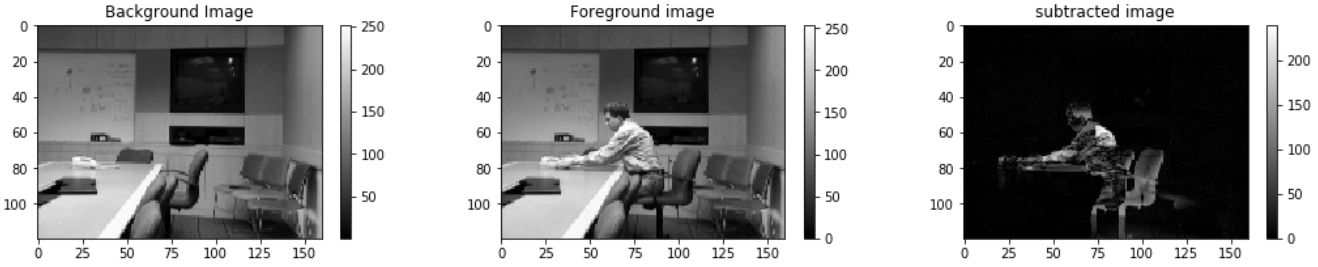


In [8]:

```
f, subplts = plt.subplots(figsize = (18, 3), ncols = 3)

background = subplts[0].imshow(image, cmap="gray")
f.colorbar(background, ax = subplts[0])
subplts[0].set_title("Background Image")

foreground = subplts[1].imshow(image2, cmap="gray")
f.colorbar(foreground, ax= subplts[1])
subplts[1].set_title("Foreground image")

subtracted = subplts[2].imshow(image3, cmap="gray")
f.colorbar(subtracted, ax= subplts[2])
subplts[2].set_title("subtracted image")
```

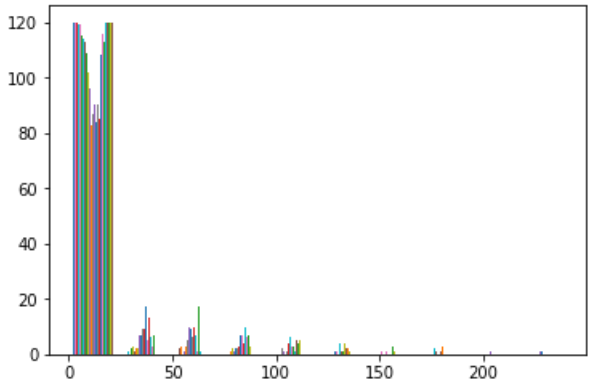
Out[8]:

```
Text(0.5,1,'subtracted image')
```



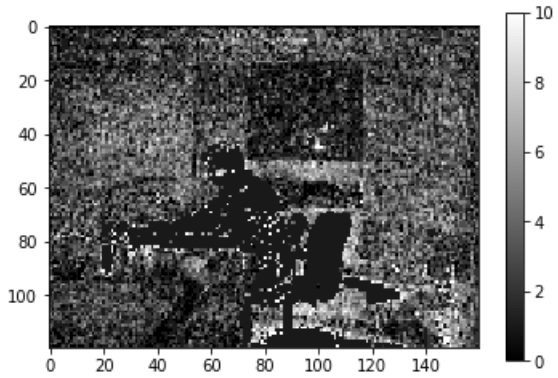
In [9]:

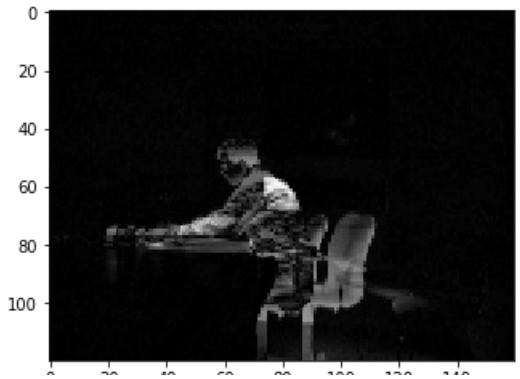
```
plt.hist(image3)
plt.show()
```

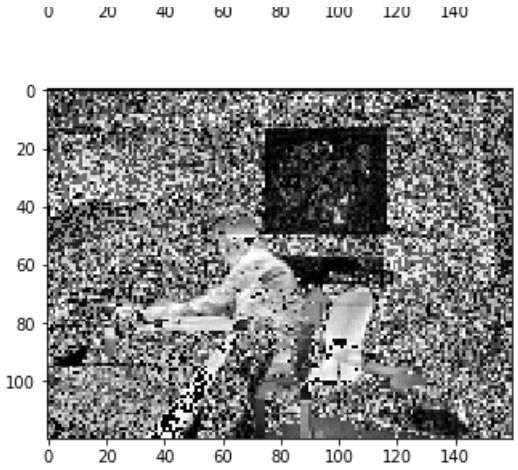


In [10]:

```
mask = np.array(image3)
mask[mask > 10] = 1
plt.imshow(mask, cmap = "gray")
plt.colorbar()
plt.show()

bs1 = image*mask + image3

# for i in range(mask.shape[0]):
#     for j in range(mask.shape[1]):
#         print(i,j)
#         print(mask[i][j])
#         print(image[i][j])
#         bs1[i][j] = mask[i][j]*image[i][j]

plt.imshow(image3, cmap="gray")
plt.show()
plt.imshow(bs1, cmap="gray")
plt.show()
```







In [11]:

```
def get_ravel(s):
    ravel = s[0].ravel()
    index = 1;

    while(index < len(s)):
        ravel = np.append(ravel, s[index])
        index = index+1

    return ravel
```

In [12]:

```
def get_dwt_coeff(img, level = 2):

    c, s = pywt.dwt2(img, 'haar')
    coeff = get_ravel(s)
    level = level - 1

    while(level > 0):
        c, s = pywt.dwt2(c, 'haar')
        coeff = np.append( get_ravel(s), coeff)
        level = level-1

    coeff = np.append(c.ravel(), coeff)
    return coeff
```

In [13]:

```
import pywt
org_coeff = get_dwt_coeff(image)
test_coeff = get_dwt_coeff(image2)
bg_coeff = get_dwt_coeff(image3)

org_coeff.shape
```

Out[13]:

```
(19200,)
```

In [14]:

```
#sparsity of matrics

org_k = np.count_nonzero(org_coeff)
test_k = np.count_nonzero(test_coeff)

bg_k = np.count_nonzero(bg_coeff)

print(org_k)
print(test_k)
print(bg_k)
```

```
17977
18028
17737
```

In [15]:

```
import math
P = np.count_nonzero(image3)

b = np.array([[org_k],[test_k], [bg_k]])
A = np.array([[N*math.log(N), N], [N*math.log(N) , N], [P*math.log(P), P]])

b_ = np.matmul(np.transpose(A), b)
A_ = np.matmul(np.transpose(A), A)

lamda = np.matmul(np.linalg.inv(A_), b_)
print(b)
print(A)
print(lamda)

approx_k = np.ceil(np.matmul(A, lamda))
print(approx_k)
```

```
[[17977]
 [18028]
 [17737]]
[[189363.1787139   19200.        ]
 [189363.1787139   19200.        ]
 [172558.03203246  17647.        ]]
[[-0.79993151]
 [ 8.82708716]]
[[18003.]
 [18003.]
 [17738.]]
```

In [16]:

```
# Measurment matrix size M

org_m = math.ceil(approx_k[0]*math.log(N/approx_k[0]))
test_m = math.ceil(approx_k[1]*math.log(N/approx_k[1]))
bg_m = math.ceil(approx_k[2]*math.log(N/approx_k[2]))

print(org_m)
print(test_m)
print(bg_m)
```

```
1159
1159
1405
```

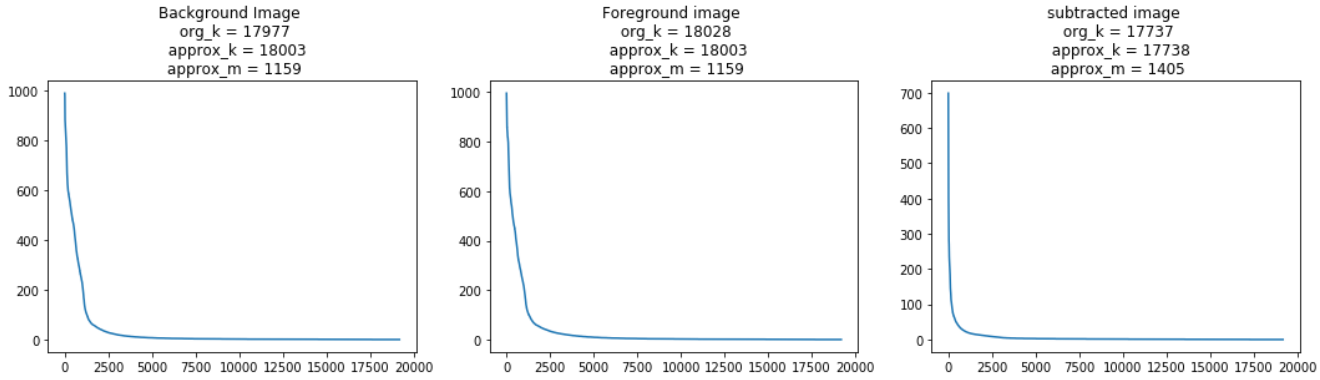
Measurment matrix size M org_m = math.ceil(org_k*math.log(N/org_k)) test_m = math.ceil(test_k*math.log(N/test_k)) bg_m = math.ceil(bg_k*math.log(N/bg_k)) print(org_m) print(test_m) print(bg_m)

In [17]:

```
print("N = "+str(N))
print("P = "+str(P))

print("lambda_0 = "+str(lamda[0]))
print("lambda_1 = "+str(lamda[1]))

f, subplts = plt.subplots(figsize = (18, 4), ncols = 3)

background = subplts[0].plot(-np.sort(-np.abs(org_coeff)))
subplts[0].set_title("Background Image \n org_k = "+str(org_k)+"\n  approx_k =
"+str(math.ceil(approx_k[0]))+"\n approx_m = "+str(org_m))

foreground = subplts[1].plot(-np.sort(-np.abs(test_coeff)))
subplts[1].set_title("Foreground image \n org_k = "+str(test_k)+"\n  approx_k = "+str(math.ceil(app
rox_k[1]))+"\n approx_m = "+str(test_m))

subtracted = subplts[2].plot(-np.sort(-np.abs(bg_coeff)))
subplts[2].set_title("subtracted image \n org_k = "+str(bg_k)+"\n  approx_k =
```

```
"+str(math.ceil(approx_k[2]))+"\n approx_m = "+str(bg_m))
plt.show()
```

```
N = 19200
P = 17647
lambda_0 = [-0.79993151]
lambda_1 = [8.82708716]
```



In [18]:

```
org_coef_s = -np.sort(-np.abs(org_coeff))
test_coef_s = -np.sort(-np.abs(test_coeff))
bg_coef_s = -np.sort(-np.abs(bg_coeff))

org_limit = org_coef_s[org_m]
test_limit = test_coef_s[test_m]
bg_limit = bg_coef_s[bg_m]
```

In [19]:

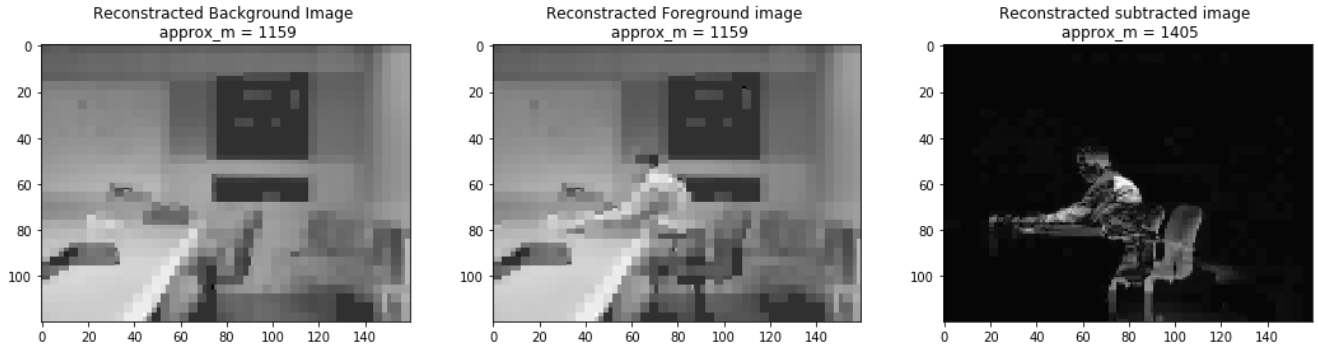
```
def get_compressed(img, limit):
    c, s1, s2 = pywt.wavedec2(img, 'haar', level = 2)

    c[np.abs(c) < limit] = 0

    for i in range(len(s1)):
        s1[i][np.abs(s1[i]) < limit] = 0
        s2[i][np.abs(s2[i]) < limit] = 0

    return c, s1, s2
```

In [20]:

```
org_c, org_s1, org_s2 = get_compressed(image, org_limit)
test_c, test_s1, test_s2 = get_compressed(image2, test_limit)
bg_c, bg_s1, bg_s2 = get_compressed(image3, bg_limit)
```

In [21]:

```
#Reconstruction
recon_org = pywt.waverec2( [org_c, org_s1, org_s2], 'haar')
recon_test = pywt.waverec2( [test_c, test_s1, test_s2], 'haar')
recon_bg = pywt.waverec2( [bg_c, bg_s1, bg_s2], 'haar')

f, subplts = plt.subplots(figsize = (18, 4), ncols = 3)

background = subplts[0].imshow(recon_org, cmap = "gray")
subplts[0].set_title("Reconstracted Background Image \n approx_m = "+str(org_m))

foreground = subplts[1].imshow(recon_test, cmap = "gray")
subplts[1].set_title("Reconstracted Foreground image \n approx_m = "+str(test_m))

subtracted = subplts[2].imshow(recon_bg, cmap = "gray")
subplts[2].set_title("Reconstracted subtracted image \n approx_m = "+str(bg_m))
plt.show()
```

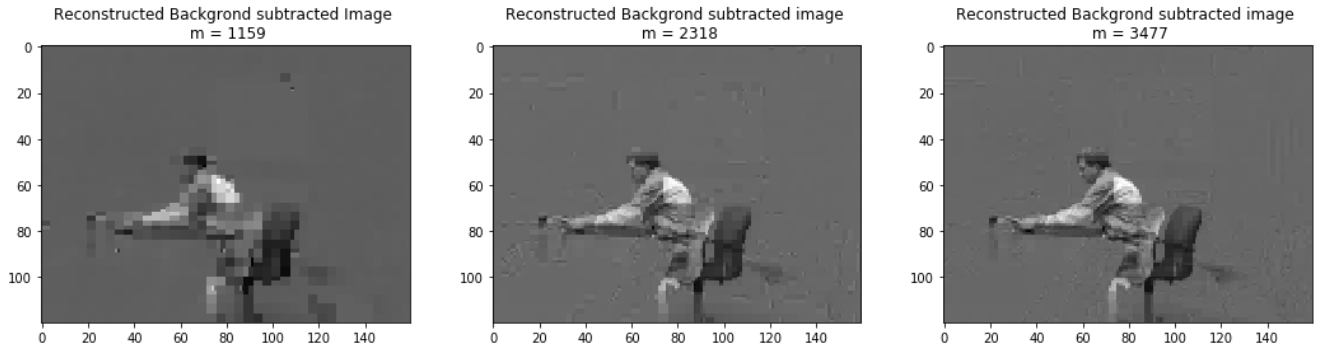


In [22]:

```
#from compressed samples of background and test images

def get_subfrom_samples( test_image, background, m):
    org_limit = org_coef_s[m]
    test_limit = test_coef_s[m]

    org_c, org_s1, org_s2 = get_compressed(background, org_limit)
    test_c, test_s1, test_s2 = get_compressed(test_image, test_limit)

    bg_s_c = test_c - org_c
    bg_s_s1 = [ test_s1[i] - org_s1[i] for i in range(3) ]
    bg_s_s2 = [ test_s2[i] - org_s2[i] for i in range(3) ]

    return pywt.waverec2( [bg_s_c, bg_s_s1, bg_s_s2], 'haar')
```

In [23]:

```
f, subplts = plt.subplots(figsize = (18, 4), ncols = 3)

background = subplts[0].imshow(get_subfrom_samples(image2, image, org_m), cmap = "gray")
subplts[0].set_title("Reconstructed Backgrond subtracted Image \n m = "+str(org_m))

foreground = subplts[1].imshow(get_subfrom_samples(image2, image, 2*org_m), cmap = "gray")
subplts[1].set_title("Reconstructed Backgrond subtracted image \n m = "+str(2*org_m))

subtracted = subplts[2].imshow(get_subfrom_samples(image2, image, 3*org_m), cmap = "gray")
subplts[2].set_title("Reconstructed Backgrond subtracted image \n m = "+str(3*org_m))
plt.show()
```



import os path = "D:\\computer vision\\MovedObject" video_path = "D:\\computer vision\\MovedObject_bg_cs" video_name = "D:\computer vision\MovedObject\\bs_cs.avi" images = [x for x in os.listdir(path) if x.endswith(".bmp")] images = images[600:900] org_m = 7*1159 background = cv2.imread("D:\\computer vision\\MovedObject\\b00001.bmp", 0) background_coeff = get_dwt_coeff(background) background_coef_s = -np.sort(-np.abs(background_coeff)) background_limit = background_coef_s[org_m] back_c, back_s1, back_s2 = get_compressed(background, background_limit) height, width = background.shape index = 1 video = cv2.VideoWriter(video_name, -1, 10, (width,height)) for image in images: test = cv2.imread(os.path.join(path, image), 0) test_coeff = get_dwt_coeff(test) test_coef_s = -np.sort(-np.abs(test_coeff)) test_limit = test_coef_s[org_m] test_c, test_s1, test_s2 = get_compressed(test, test_limit) bg_s_c = test_c - back_c bg_s_s1 = [test_s1[i] - back_s1[i] for i in range(3)] bg_s_s2 = [test_s2[i] - back_s2[i] for i in range(3)] bg_image = pywt.waverec2([bg_s_c, bg_s_s1, bg_s_s2], 'haar') plt.imsave(os.path.join(video_path,str(index)+".jpg"), bg_image, cmap='gray') video.write(cv2.imread(os.path.join(video_path,str(index)+".jpg"))) index = index+1; cv2.destroyAllWindows() video.release()img = cv2.imread(os.path.join(video_path,"1.jpg")) print(img.shape) height, width, layers = img.shape video = cv2.VideoWriter(video_name, -1, 10, (width,height)) index = 0 for img in os.listdir(video_path): print(img) image = cv2.imread(os.path.join(video_path, img))

```
video.write(image) index = index+1 import os alpha = 0.8 path = "D:\\computer vision\\MovedObject" video_path = "D:\\computer vision\\MovedObject_adap_bg_"+str(alpha) video_name = "D:\computer vision\MovedObject\\adap_bs_cs_"+str(alpha)+".avi" if not os.path.exists(video_path): os.mkdir(video_path) images = [x for x in os.listdir(path) if x.endswith(".bmp")] images = images[600:900] org_m = 7*1159 background = cv2.imread("D:\\computer vision\\MovedObject\\b00001.bmp", 0) background_coeff = get_dwt_coeff(background) background_coef_s = -np.sort(-np.abs(background_coeff)) background_limit = background_coef_s[org_m] back_c, back_s1, back_s2 = get_compressed(background, background_limit) height, width = background.shape index = 1 video = cv2.VideoWriter(video_name, -1, 10, (width,height)) for image in images: test = cv2.imread(os.path.join(path, image), 0) test_coeff = get_dwt_coeff(test) test_coef_s = -np.sort(-np.abs(test_coeff)) test_limit = test_coef_s[org_m] test_c, test_s1, test_s2 = get_compressed(test, test_limit) bg_s_c = test_c - back_c bg_s_s1 = [ test_s1[i] - back_s1[i] for i in range(3) ] bg_s_s2 = [ test_s2[i] - back_s2[i] for i in range(3) ] bg_image = pywt.waverec2( [bg_s_c, bg_s_s1, bg_s_s2], 'haar') plt.imsave(os.path.join(video_path,str(index)+".jpg"), bg_image, cmap='gray') back_recon = pywt.waverec2( [back_c, back_s1, back_s2], 'haar') test_recon = pywt.waverec2( [test_c, test_s1, test_s2], 'haar') bs_est = test_recon - back_recon bs_est_coeff = get_dwt_coeff(bs_est) bs_est_coef_s = -np.sort(-np.abs(bs_est_coeff)) bs_est_limit = bs_est_coef_s[org_m] bs_est_c, bs_est_s1, bs_est_s2 = get_compressed(bs_est, bs_est_limit) back_c = alpha*(test_c - bs_est_c) + (1-alpha)*(back_c) back_s1 = [ alpha*(test_s1[i] - bs_est_s1[i]) + (1-alpha)*(back_s1[i]) for i in range(3) ] back_s2 = [ alpha*(test_s2[i] - bs_est_s2[i]) + (1-alpha)*(back_s2[i]) for i in range(3) ] video.write(cv2.imread(os.path.join(video_path,str(index)+".jpg"))) index = index+1; cv2.destroyAllWindows() video.release()
```