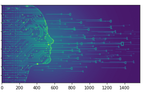
In [1]:
```
import numpy as np
from matplotlib import pyplot as plt
import cv2
```

In [2]:
```
original_image = cv2.imread('foundationscomputerscience.jpg', 0)
plt.imshow(original_image)
```

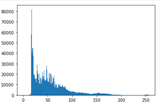
Out[2]:
```
<matplotlib.image.AxesImage at 0x7fbc31767410>
```



In [3]:
```
image_pixels = np.array([original_image])
image_pixels
```

Out[3]:
```
array([[[25, 22, 26, ..., 16, 16, 16],
        [27, 24, 26, ..., 16, 16, 16],
        [27, 24, 23, ..., 16, 16, 16],
        ...,
        [21, 29, 22, ..., 16, 16, 16],
        [27, 22, 19, ..., 16, 16, 16],
        [76, 52, 46, ..., 16, 16, 16]]], dtype=uint8)
```

In [4]:
```
plt.hist(image_pixels.ravel(), 256, [0,256])
plt.show()
```



In [ ]:
```
#image_1 = original_image
#plt.imshow(image_1, cmap='HdrlGn')
```

In [6]:
```
def show(image):
  plt.imshow(image, cmap = 'gray')
  plt.xticks([])
  plt.yticks([])
```

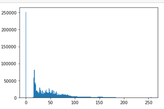
In [7]:
```
height, width = original_image.shape
img = np.zeros((height + 160, width), np.uint8)
img[80:-80, :] = original_image

plt.figure(figsize=(15,5))
plt.subplot(131)
show(img)
```

In [8]:
```
gray_pixels = np.array([img])
gray_pixels
```

Out[8]:
```
array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8)
```

In [9]:
```
plt.hist(img.ravel(), 256, [0,256])
plt.show()
```



In [10]:
```
!pip install sewar
```

```
Requirement already satisfied: sewar in /usr/local/lib/python3.7/dist-packages (0.4.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from sewar) (7.1.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from sewar) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from sewar) (1.21.5)
```

In [11]:
```
from sewar import full_ref
from skimage import measure, metrics

from PIL import Image

from google.colab.patches import cv2_imshow
```

Applying the mean filter with 3x3 kernel

In [12]:
```
mean_kernel = np.ones((3, 3), np.float32) / 25
```

In [13]:
```
convolution_mean = cv2.filter2D(img, -1, mean_kernel, borderType = cv2.BORDER_CONSTANT)
plt.figure(figsize=(15, 5))
show(convolution_mean)
```

In [14]:
```
rmse_skimg = metrics.normalized_root_mse(img, convolution_mean)
print('RMSE: based on scikit-image = ', rmse_skimg)
```

```
RMSE: based on scikit-image =  0.6512582223038206
```

In [15]:
```
mse_skimg = metrics.mean_squared_error(img, convolution_mean)
print('MSE: based on scikit-image = ', mse_skimg)
```

```
MSE: based on scikit-image =  1586.5428254881372
```

In [16]:
```
psnr_skimg = metrics.peak_signal_noise_ratio(img, convolution_mean, data_range=None)
print('PSNR: based on scikit-image = ', psnr_skimg)
```

```
PSNR: based on scikit-image =  16.126285613793765
```

In [17]:
```
from skimage.metrics import structural_similarity as ssim

ssim_skimg = ssim(img, convolution_mean, data_range = img.max() - img.min(), multichannel = True)
print('SSIM: based on scikit-image = ', ssim_skimg)
```

```
SSIM: based on scikit-image =  0.5997347078882577
```

Applying the mean filter with 5x5 kernel

In [18]:
```
mean_kernel = np.ones((5, 5), np.float32) / 25
```

In [19]:
```
convolution_mean = cv2.filter2D(img, -1, mean_kernel, borderType = cv2.BORDER_CONSTANT)
plt.figure(figsize=(15, 5))
show(convolution_mean)
```

In [20]:
```
rmse_skimg = metrics.normalized_root_mse(img, convolution_mean)
print('RMSE: based on scikit-image = ', rmse_skimg)
```

```
RMSE: based on scikit-image =  0.19344577109999714
```

In [21]:
```
mse_skimg = metrics.mean_squared_error(img, convolution_mean)
print('MSE: based on scikit-image = ', mse_skimg)
```

```
MSE: based on scikit-image =  139.97930755299882
```

In [22]:
```
psnr_skimg = metrics.peak_signal_noise_ratio(img, convolution_mean, data_range=None)
print('PSNR: based on scikit-image = ', psnr_skimg)
```

```
PSNR: based on scikit-image =  26.670165200449606
```

In [23]:
```
from skimage.metrics import structural_similarity as ssim

ssim_skimg = ssim(img, convolution_mean, data_range = img.max() - img.min(), multichannel = True)
print('SSIM: based on scikit-image = ', ssim_skimg)
```

```
SSIM: based on scikit-image =  0.8875961528181259
```

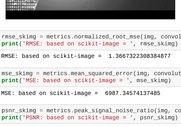
In [23]:
```

```

Applying the mean filter with 8x8 kernel

In [24]:
```
mean_kernel = np.ones((8, 8), np.float32) / 25
```

In [25]:
```
convolution_mean = cv2.filter2D(img, -1, mean_kernel, borderType = cv2.BORDER_CONSTANT)
plt.figure(figsize=(15, 5))
show(convolution_mean)
```



In [26]:
```
rmse_skimg = metrics.normalized_root_mse(img, convolution_mean)
print('RMSE: based on scikit-image = ', rmse_skimg)
```

```
RMSE: based on scikit-image =  1.3667322308384877
```

In [27]:
```
mse_skimg = metrics.mean_squared_error(img, convolution_mean)
print('MSE: based on scikit-image = ', mse_skimg)
```

```
MSE: based on scikit-image =  6987.34574137485
```

In [28]:
```
psnr_skimg = metrics.peak_signal_noise_ratio(img, convolution_mean, data_range=None)
print('PSNR: based on scikit-image = ', psnr_skimg)
```

```
PSNR: based on scikit-image =  9.687681277242984
```

In [29]:
```
from skimage.metrics import structural_similarity as ssim

ssim_skimg = ssim(img, convolution_mean, data_range = img.max() - img.min(), multichannel = True)
print('SSIM: based on scikit-image = ', ssim_skimg)
```

```
SSIM: based on scikit-image =  0.5573752691359672
```

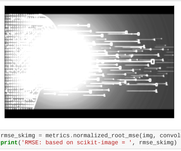
In [29]:
```

```

Applying the mean filter with 10x10 kernel

In [30]:
```
mean_kernel = np.ones((10, 10), np.float32) / 25
```

In [31]:
```
convolution_mean = cv2.filter2D(img, -1, mean_kernel, borderType = cv2.BORDER_CONSTANT)
plt.figure(figsize=(15, 5))
show(convolution_mean)
```



In [32]:
```
rmse_skimg = metrics.normalized_root_mse(img, convolution_mean)
print('RMSE: based on scikit-image = ', rmse_skimg)
```

```
RMSE: based on scikit-image =  2.060715437808864
```

In [33]:
```
mse_skimg = metrics.mean_squared_error(img, convolution_mean)
print('MSE: based on scikit-image = ', mse_skimg)
```

```
MSE: based on scikit-image =  15884.787517691164
```

In [34]:
```
psnr_skimg = metrics.peak_signal_noise_ratio(img, convolution_mean, data_range=None)
print('PSNR: based on scikit-image = ', psnr_skimg)
```

```
PSNR: based on scikit-image =  6.120909509009393
```

In [35]:
```
from skimage.metrics import structural_similarity as ssim

ssim_skimg = ssim(img, convolution_mean, data_range = img.max() - img.min(), multichannel = True)
print('SSIM: based on scikit-image = ', ssim_skimg)
```

```
SSIM: based on scikit-image =  0.40245689065915996
```

In [ ]:
```
!jupyter nbconvert --to html Seatwork_1.ipynb
```