影像處理 HW1　　311551159 謝岑

A. Intensity Transformation Using Gamma Correction

一、 實作目標

本實作任務透過 gamma correction 對彩色圖像進行 intensity transformation，比較不同數值的 gamma 所產生的差異。此外，也另外實驗並觀察若改為同時對 RGB channels 進行 gamma correction 時會產生的影響。

二、 方法介紹

Gamma correction 為一種 intensity transformation 的方法，其公式為：

$$s = c\, r^{\gamma}$$

其中r為 input intensity，s為 output intensity，此兩值介於 0 至 1 之間，因此實作時通常會對圖像的 pixel value 正規化後進行運算；c為一常數值，通常設為 1；γ為 gamma 值，當γ越大，傾向使圖像中 intensity 大的部分對比減少，intensity 小的部分對比增加，而當γ越小，傾向使 intensity 大的部分對比增加，intensity 小的部分對比減少。

一般來說，若要對彩色圖像以 gamma correction 進行 intensity transformation，需針對圖像的 luminance channel 進行運算，其餘 channels 的數值則不變。

三、實作過程、結果與討論

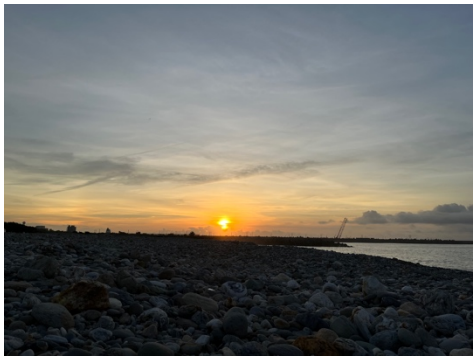
Gamma correction 可以改變圖像的對比度，因此在這裡選用了兩張拍攝光線不佳的圖像作為實驗對象：

圖一

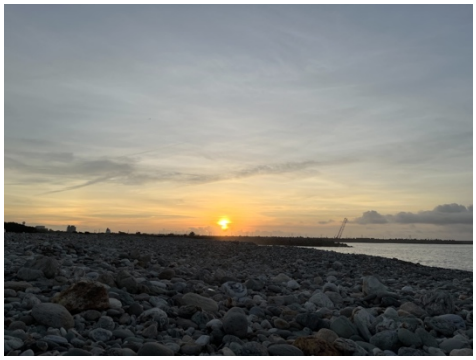
圖二

圖一逆光拍攝，造成主體人物背光，圖二為日出時分拍攝的影像，因此光線不足。分別對圖一、圖二進行 gamma correction 後的結果如下：

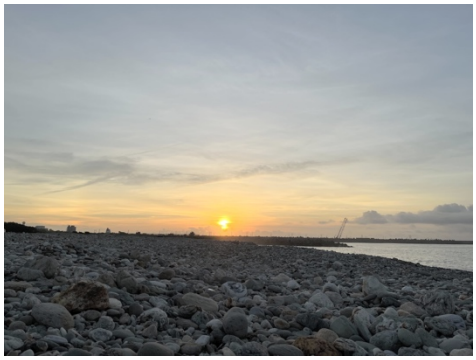
原圖

$\gamma = 0.8$

$\gamma = 0.6$

$\gamma = 0.4$


由結果可以發現，隨著 gamma 值變小，圖一原先光線不足處的亮度對比被加大，讓中間背光的人物清晰起來，原先亮度高的背景天空則對比度減少；而圖二原先光線不足的石頭海岸逐漸變得顆顆分明，但同時原先亮度充足的太陽則仍舊保持明亮。

除了僅對 luminance channel 進行 gamma correction 之外，也另外嘗試同時對 RGB channels 進行 gamma correction，並比較兩者：

luminance channel, $\gamma = 0.4$

RGB channels, $\gamma = 0.4$

由結果可以發現，同時對 RGB channels 進行 gamma correction 時，可能會使產生的圖像出現顏色的偏差，例如左圖海岸上有兩顆較大的深灰色的石頭，在右圖卻呈現紅棕色，左圖橘黃的日出在右圖則顏色偏白。因為同時對 RGB channels 進行 gamma correction 時，有機會出現 color shift，因此彩色圖像的 intensity transformation 會選擇僅對 luminance channel 進行 gamma correction 而非 RGB 三個 channels。

B. Denoising

一、實作目標

本實作任務分別以 gaussian filter 與 median filter 對灰階圖像進行 denoising，實作目標分為以下三個部分：

1. 以 gaussian filter 進行 denoising，試驗不同 filter 大小和不同標準差值對結果所產生的影響。
2. 以 median filter 進行 denoising，試驗不同 filter 大小對結果所產生的影響。
3. 比較 gaussian filter 與 median filter 的 denoising 效果。

二、方法介紹

為了進行 denoising 的實驗，首先需要對原圖加入 noise，這裡使用到的 noise 種類包含 gaussian noise 與 salt-and-pepper noise：

- Gaussian noise：

  Gaussian noise 是一種具常態分佈機率密度函數的 noise。給定分佈的平均值與標準差，即可以決定一常態分佈。實作在圖像中加入 gaussian noise，需對

圖像各個的 pixel values 正規化，再與從常態分佈中任意抽取的隨機值相加後反正規化，即可生成具有 gaussian noise 的圖像。

- Salt-and-pepper noise：

  Salt-and-pepper noise 是一種在圖像中隨機出現的黑點或白點雜訊。隨機選擇圖像中的任意數量的 pixels 並將其值改為 0 或 255，即可對圖像添加 salt-and-pepper noise。

實作中使用到的 filter 包含 gaussian filter 與 median filter，兩者進行 filtering 的方法分述如下：

- Gaussian filter：

  Gaussian filter 是一種 smoothing filter，透過 gaussian distribution 來計算圖像中每個 pixel 的變換。Gaussian filter 以整數來逼近一 gaussian distribution，進行卷積時會對中間 pixel 鄰近的 pixel values 計算加權平均後得到 filter 後的值。

  在二維空間下，gaussian distribution 的函數定義為：

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/(2\sigma^2)}$$

  以 filter 大小 3x3，標準差$\sigma=1$為例，定義中心點為原點$(0,0)$，將每個座標值經過 gaussian function 並正規化後，即可得到逼近 gaussian distribution 的 gaussian filter，圖示如下：

(-1, -1)	(0, -1)	(1, -1)
(-1, 0)	(0, 0)	(1, 0)
(-1, 1)	(0, 1)	(1, 1)

$G(x,y)$, normalize →

0.05855	0.09653	0.05855
0.09653	0.15915	0.09653
0.05855	0.09653	0.05855

  以此 gaussian filter 對圖像做卷積，即可得到完成 gaussian filtering 後的結果。

- Median filter：

  Median filter 為一種非線性的 filter 技術。Filter 對圖像進行卷積的過程中，逐一找出 filter 涵蓋範圍內所有數值的 median，並作為中間 pixel 的數值，即可得到完成 median filtering 後的圖像。

  舉例而言，如果 filter 涵蓋的範圍內的 pixel values 如下：

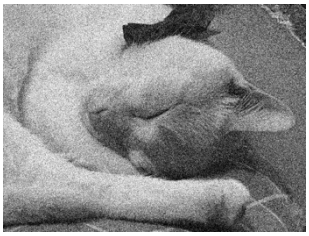
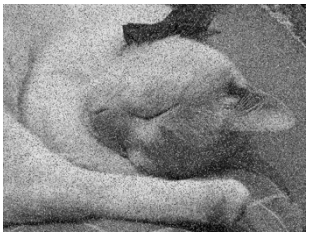
2	2	3
3	10	1
3	5	5

  則應取 3 作為中間 pixel 的數值。

另外在卷積過程中，當 filter 涵蓋範圍超過圖像邊界時，會使用到在邊界外填充 0 的 padding 技術。

三、實作過程、結果與討論

進行 denoising 之前，需對實驗圖像添加 noise。這裡 gaussian noise 取自平均為 0、標準差 0.2 的常態分佈。以下為實驗圖像的原圖與加上 noise 之後的結果：

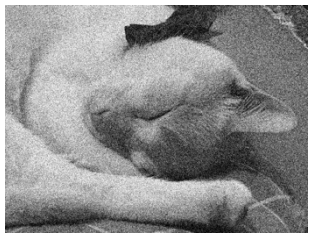
	圖三	圖四	圖五
原圖			
Gaussian noise			
Salt-and-pepper noise			

首先以添加 gaussian noise 的圖五為對象，分別以三種大小的 filter 與三種標準差（σ）組合成的九種的 gaussian filter 對圖像做 denoising，結果如下：

	7x7	11x11	13x13
$\sigma = 1$			
$\sigma = 2$			
$\sigma = 5$			

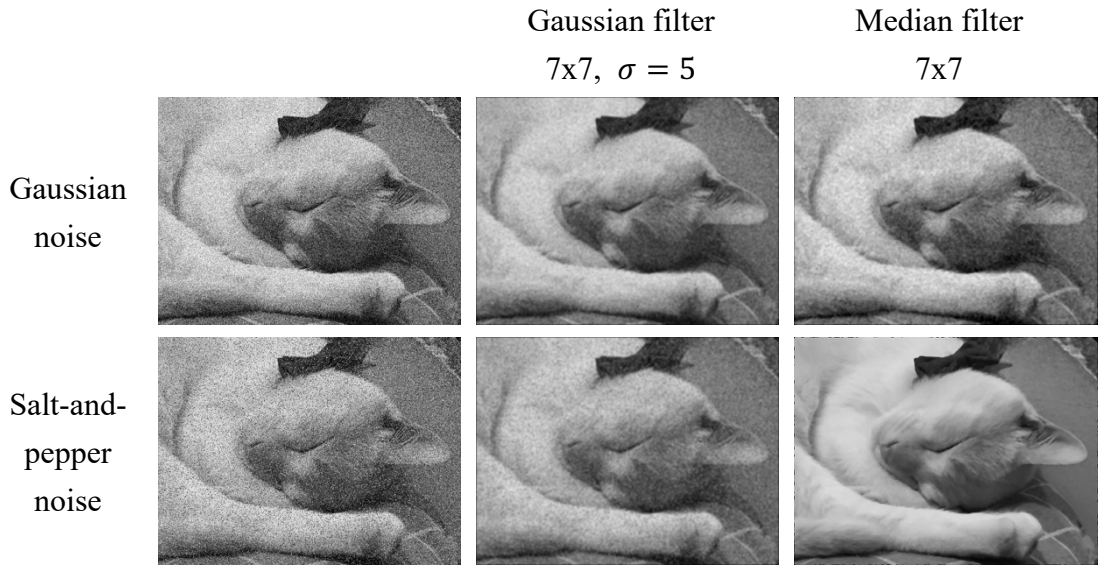
由上面的結果可以發現，對於 gaussian filter 而言，filter 的大小與 σ 大小皆對 denoising 的最終結果有影響。Filter 尺寸越大，表示決定目標 pixel 的數值時考慮的範圍越大，因此 denoising 的效果越好，但同時也會讓圖像變得模糊；而隨著 σ 越大，表示 filter 範圍之下離目標 pixel 較遠的值權重跟著增加，因此也會傾向使平滑化越明顯。

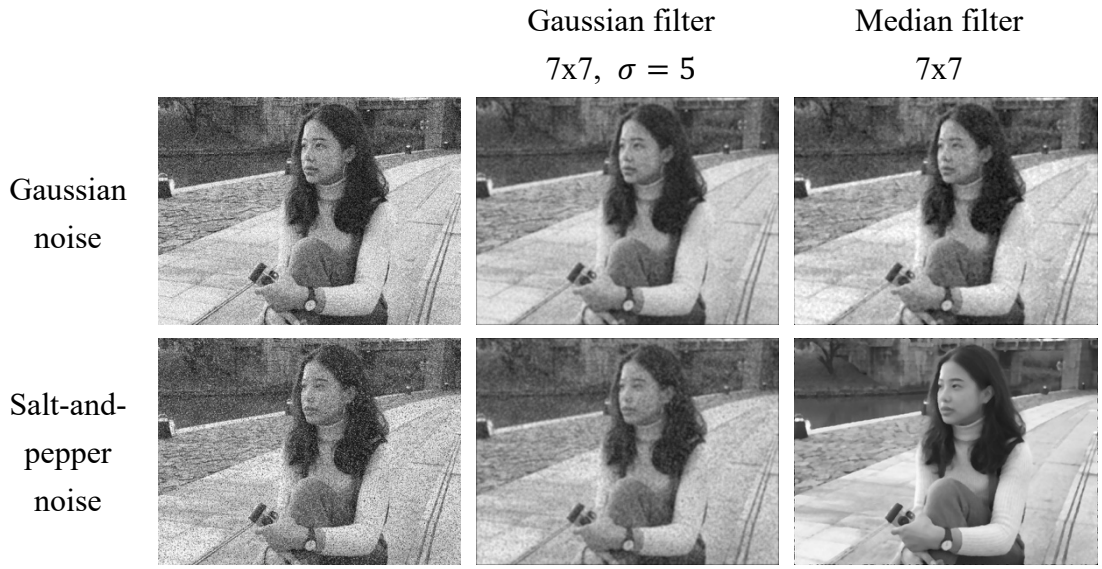
接著比較使用不同 filter 大小的 median filter 對 denoising 效果的差異，這裡實驗的對象為添加 gaussian noise 的圖三、圖四與圖五：

原圖			
Gaussian noise			
3x3			
5x5			
7x7			

比較各個 denoising 的結果，可以得知隨著 median filter 的 filter 大小越大，會使圖像 denoising 的效果越好，但也同時會讓圖像變得模糊。

最後比較 gaussian filter 與 median filter 對圖像進行 denoising 的結果差異。這裡使用的 gaussian filter 為大小 7x7、$\sigma = 5$，median filter 為大小 7x7，實驗對象為分別添加 gaussian noise 和 salt-and-pepper noise 的圖四、圖五：





針對 gaussian noise，gaussian filter 和 median filter 在 denoising 效果上沒有太明顯的差異，然而針對 salt-and-pepper noise，兩者得到的效果差異顯著。

Gaussian filter 沒有辦法有好的 denoising 效果，是因為 gaussian filter 屬於 averaging filter 的一種，在計算目標 pixel 數值時是取鄰近 pixel 值的加權平均，而因為添加 salt-and-pepper noise 的圖像具有多個極端的 pixel value，容易影響 averaging filter 的的計算與取值。

Median filter 對加了 salt-and-pepper noise 的圖像進行 denoising，則可以清除大部分的 noise。從運作方式來看，median filter 會選取 filter 內數值的 median，因此極端值不會影響取值，filter 容易選取到不受 noise 影響的 pixel value，從而使輸出的圖像有好的 denoising 結果。

In [1]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import random
import math
```

In [2]:
```
# read files
img_1 = cv2.imread('images/IMG-1.jpg')
img_2 = cv2.imread('images/IMG-2.jpg')
img_3 = cv2.imread('images/IMG-3.jpg')
img_4 = cv2.imread('images/IMG-4.jpg')
img_5 = cv2.imread('images/IMG-5.jpg')
```

In [3]:
```
def show_image(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # change color channels
    plt.imshow(image)
    plt.show()

def to_grayscale(image):
    image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    return image
```

In [4]:
```
def gamma_corr_luminance(image, gamma):
    yuv_image = cv2.cvtColor(image, cv2.COLOR_BGR2YUV)
    for i in range(yuv_image.shape[0]):
        for j in range(yuv_image.shape[1]):
            value = yuv_image[i][j][0]
            yuv_image[i][j][0] = (value/255)**gamma * 255 # normalization
    new_image = cv2.cvtColor(yuv_image, cv2.COLOR_YUV2BGR)
    return new_image

def gamma_corr_rgb(image, gamma):
    # formula: output_intensity = c * input_intensity**gamma
    new_image = np.zeros(image.shape, dtype='uint8')
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            for c in range(3):
                value = image[i][j][c]
                new_image[i][j][c] = (value/255)**gamma * 255
    return new_image
```

In [5]:
```
def gaussian_noise(image, mean, sigma):
    noise = np.random.normal(mean, sigma, image.shape)
    new_image = np.zeros(image.shape, dtype='uint8')
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            value = image[i][j] / 255
            value = value + noise[i][j]
            if value > 1: value = 1
            elif value < 0: value = 0
            new_image[i][j] = value * 255
    return new_image
```

In [6]:
```
def sp_noise(image):
    n_pixels = int(image.shape[0] * image.shape[1] / 10)
```

```
    new_image = image.copy()
    for i in range(n_pixels):
        y_index = random.randint(0, image.shape[0]-1)
        x_index = random.randint(0, image.shape[1]-1)
        new_image[y_index][x_index] = 255
    for i in range(n_pixels):
        y_index = random.randint(0, image.shape[0]-1)
        x_index = random.randint(0, image.shape[1]-1)
        new_image[y_index][x_index] = 0
    return new_image
```

In [7]:

```
def gaussian_filter(image, mask_size, sigma):
    g_filter = []
    total = 0
    for i in range(mask_size):
        row_list = []
        for j in range(mask_size):
            x = -mask_size//2 + j + 1
            y = -mask_size//2 + i + 1
            ans = ( (1 / (2*math.pi*sigma**2)) *
                    math.exp(-( (x**2 + y**2) / (2*sigma**2) )) )
            row_list.append(ans)
            total += ans
        g_filter.append(row_list)
    g_filter = np.array(g_filter)

    # normalization
    for i in range(mask_size):
        for j in range(mask_size):
            g_filter[i][j] = g_filter[i][j] / total

    new_image = np.zeros(image.shape, dtype='uint8')
    # filtering
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            new_value = 0
            for p in range(mask_size):
                for q in range(mask_size):
                    y_index = i - mask_size//2 + p
                    x_index = j - mask_size//2 + q
                    if y_index < 0 or x_index < 0: pixel_value = 0
                    elif (y_index >= image.shape[0]
                          or x_index >= image.shape[1]): pixel_value = 0
                    else: pixel_value = image[y_index][x_index]
                    new_value += pixel_value * g_filter[q][p]
            new_image[i][j] = new_value
    return new_image
```

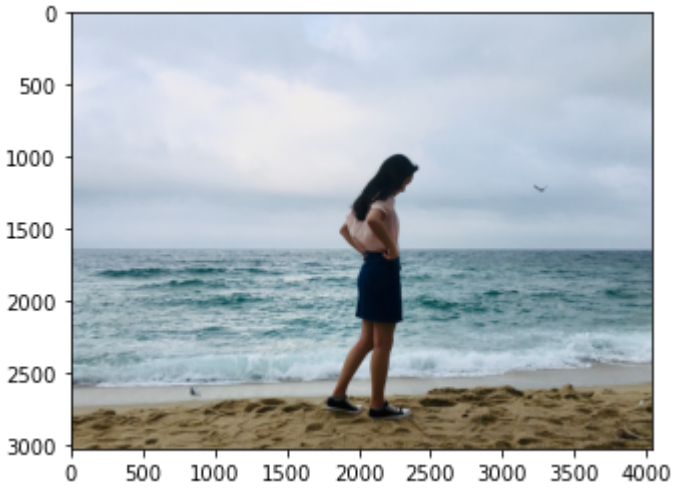
In [8]:

```
def median_filter(image, mask_size):
    new_image = np.zeros(image.shape, dtype='uint8')
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            mask_list = []
            for p in range(mask_size):
                for q in range(mask_size):
                    y_index = i - mask_size//2 + p
                    x_index = j - mask_size//2 + q
                    if y_index < 0 or x_index < 0: mask_list.append(0)
                    elif (y_index >= image.shape[0]
                          or x_index >= image.shape[1]): mask_list.append(0)
                    else: mask_list.append(image[y_index][x_index])
```

```
            mask_list.sort()
            m = mask_list[(mask_size * mask_size) // 2]
            new_image[i][j] = m
    return new_image
```

In [9]:

```
show_image(img_1)
```



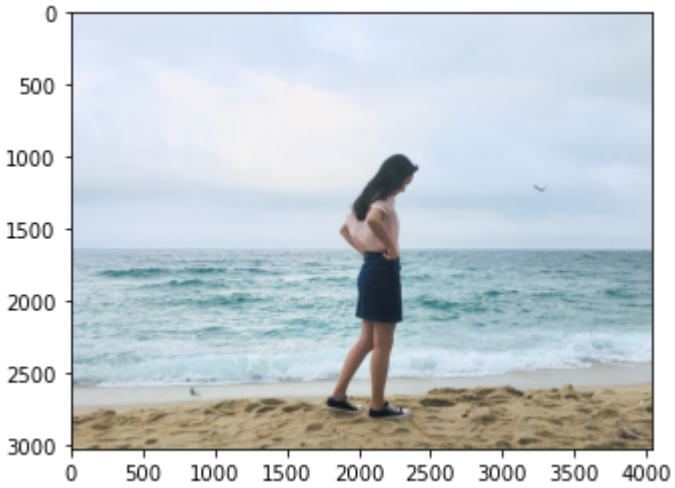
In [10]:

```
img_1_gamma_lumin = gamma_corr_luminance(img_1, 0.8)
show_image(img_1_gamma_lumin)
cv2.imwrite('result_images/IMG-1-1.jpg', img_1_gamma_lumin)
```

Out[10]: True

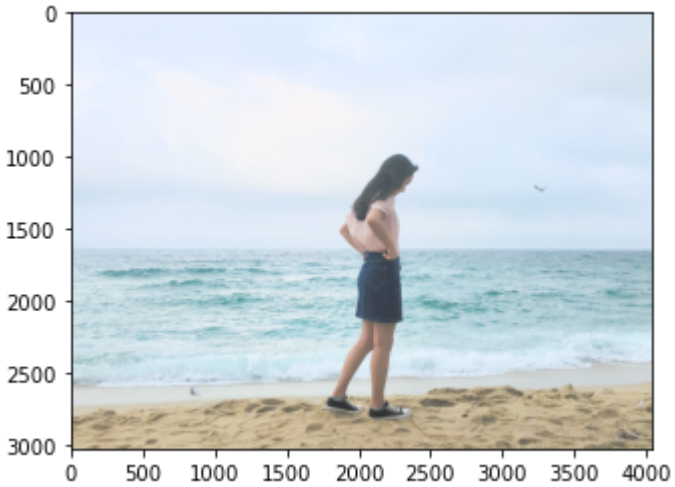
In [11]:

```
img_1_gamma_lumin = gamma_corr_luminance(img_1, 0.6)
show_image(img_1_gamma_lumin)
cv2.imwrite('result_images/IMG-1-2.jpg', img_1_gamma_lumin)
```



Out[11]: True

In [12]:
```
img_1_gamma_lumin = gamma_corr_luminance(img_1, 0.4)
show_image(img_1_gamma_lumin)
cv2.imwrite('result_images/IMG-1-3.jpg', img_1_gamma_lumin)
```



Out[12]: True

In [13]:
```
show_image(img_2)
```

In [14]:
```
img_2_gamma_lumin = gamma_corr_luminance(img_2, 0.8)
show_image(img_2_gamma_lumin)
```

```
cv2.imwrite('result_images/IMG-2-1.jpg', img_2_gamma_lumin)
```

Out[14]: True

In [15]:
```
img_2_gamma_lumin = gamma_corr_luminance(img_2, 0.6)
show_image(img_2_gamma_lumin)
cv2.imwrite('result_images/IMG-2-2.jpg', img_2_gamma_lumin)
```

Out[15]: True

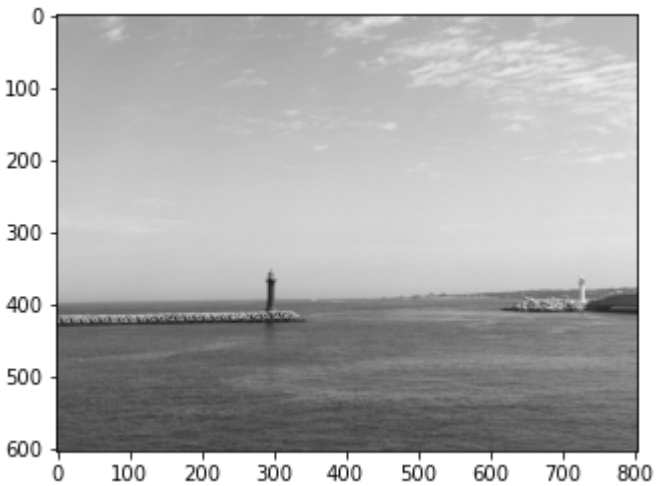
In [16]:
```
img_2_gamma_lumin = gamma_corr_luminance(img_2, 0.4)
show_image(img_2_gamma_lumin)
cv2.imwrite('result_images/IMG-2-3.jpg', img_2_gamma_lumin)
```

Out[16]: True

In [17]:
```
img_2_gamma = gamma_corr_rgb(img_2, 0.4)
show_image(img_2_gamma)
cv2.imwrite('result_images/IMG-2-4.jpg', img_2_gamma)
```

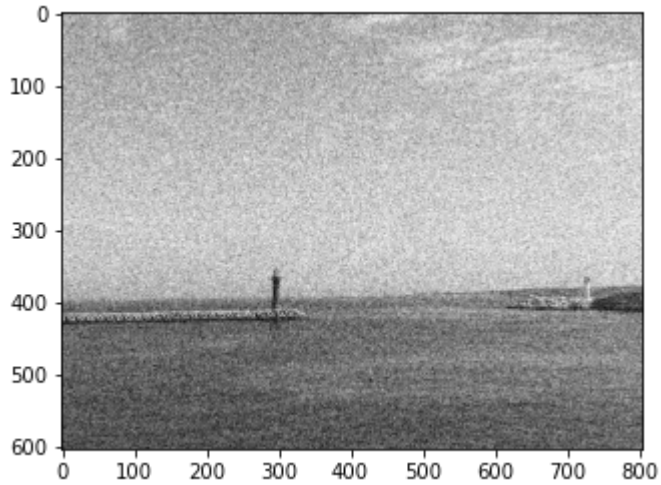
Out[17]: True

In [18]:
```
img_3_small = cv2.resize(img_3,
                         (int(img_3.shape[1]/5), int(img_3.shape[0]/5)),
                         interpolation=cv2.INTER_AREA)
img_3_gray = to_grayscale(img_3_small)
show_image(img_3_gray)
cv2.imwrite('result_images/IMG-3-gray.jpg', img_3_gray)
```



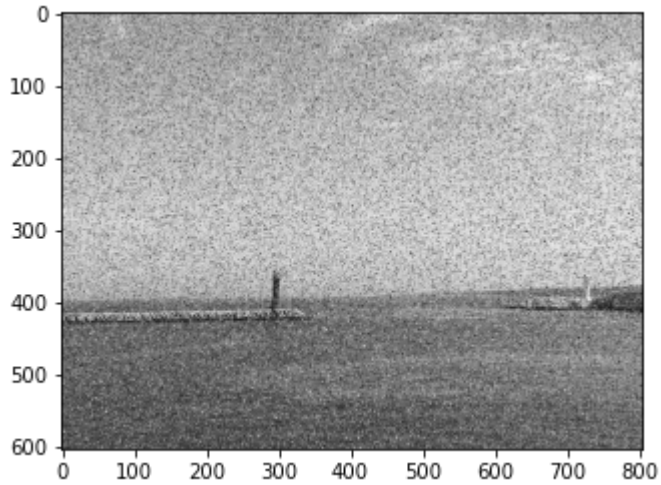
Out[18]: True

In [19]:
```
img_3_gnoise = gaussian_noise(img_3_gray, 0, 0.2)
show_image(img_3_gnoise)
cv2.imwrite('result_images/IMG-3-gnoise.jpg', img_3_gnoise)
```



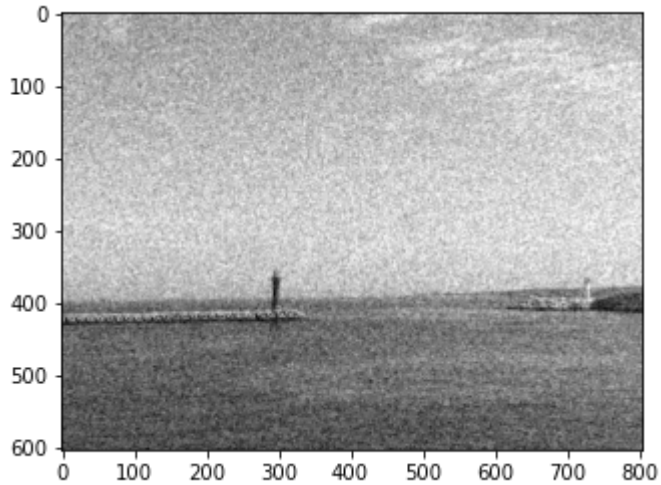
Out[19]: True

In [20]:
```
img_3_spnoise = sp_noise(img_3_gray)
show_image(img_3_spnoise)
cv2.imwrite('result_images/IMG-3-spnoise.jpg', img_3_spnoise)
```



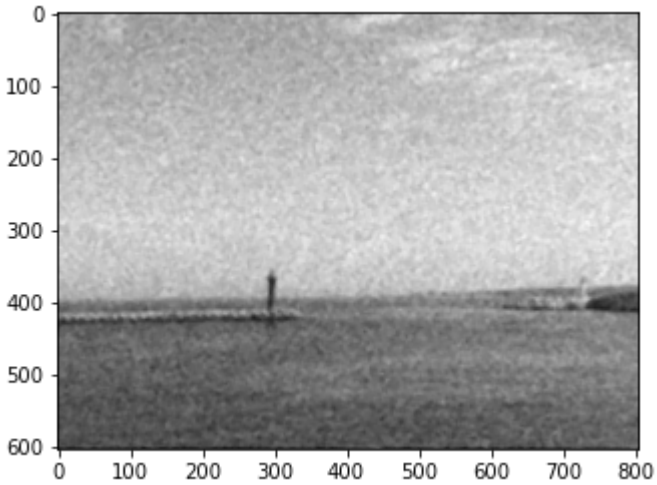
Out[20]: True

In [21]:
```
img_3_filtered = median_filter(img_3_gnoise, 3)
show_image(img_3_filtered)
cv2.imwrite('result_images/IMG-3-g-median-3.jpg', img_3_filtered)
```



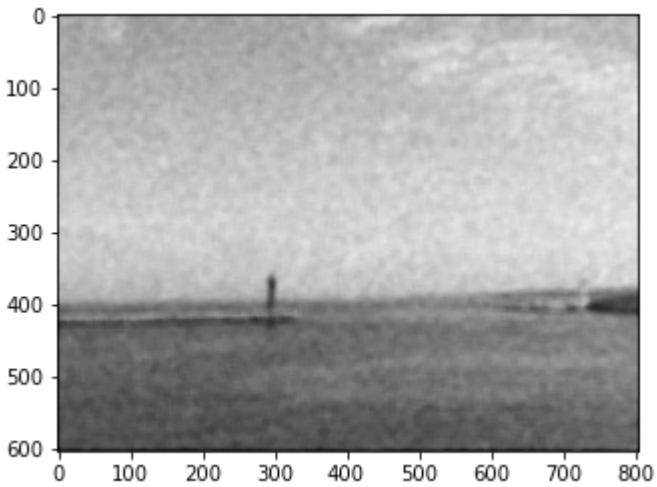
Out[21]: True

In [22]:
```
img_3_filtered = median_filter(img_3_gnoise, 7)
show_image(img_3_filtered)
cv2.imwrite('result_images/IMG-3-g-median-7.jpg', img_3_filtered)
```



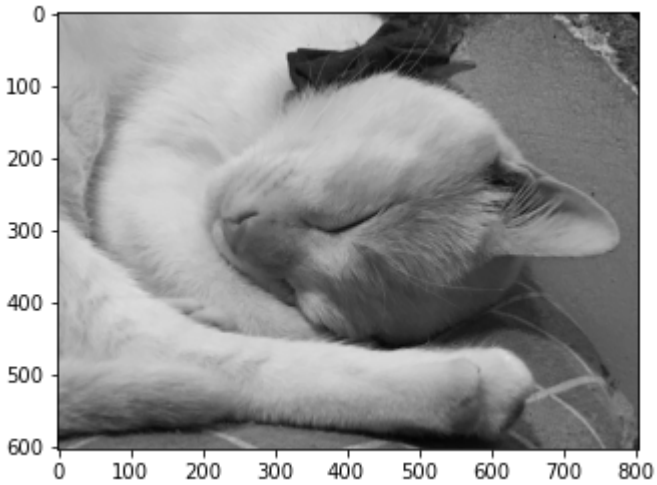
Out[22]: True

In [23]:
```
img_3_filtered = median_filter(img_3_gnoise, 11)
show_image(img_3_filtered)
cv2.imwrite('result_images/IMG-3-g-median-11.jpg', img_3_filtered)
```



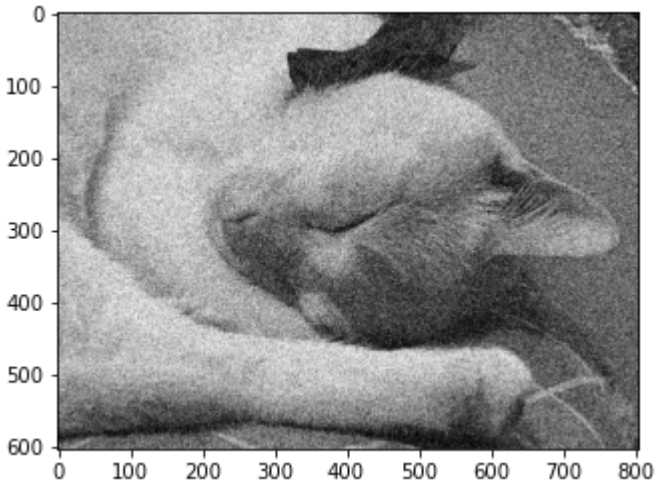
Out[23]: True

In [26]:
```
img_4_small = cv2.resize(img_4,
                         (int(img_4.shape[1]/5), int(img_4.shape[0]/5)),
                         interpolation=cv2.INTER_AREA)
img_4_gray = to_grayscale(img_4_small)
show_image(img_4_gray)
cv2.imwrite('result_images/IMG-4-gray.jpg', img_4_gray)
```



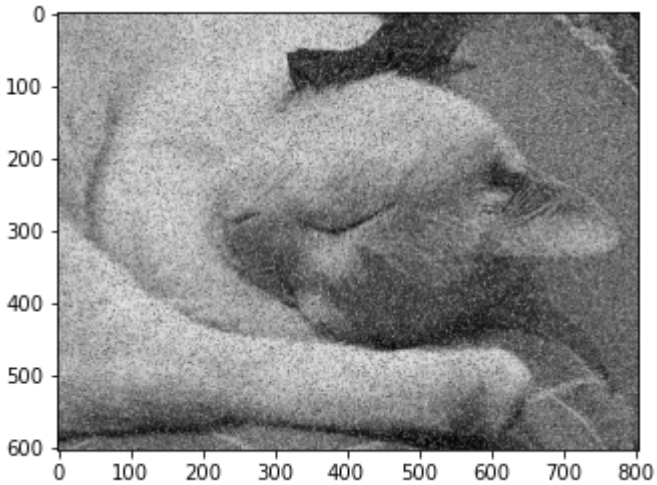
Out[26]: True

In [27]:
```
img_4_gnoise = gaussian_noise(img_4_gray, 0, 0.2)
show_image(img_4_gnoise)
cv2.imwrite('result_images/IMG-4-gnoise.jpg', img_4_gnoise)
```



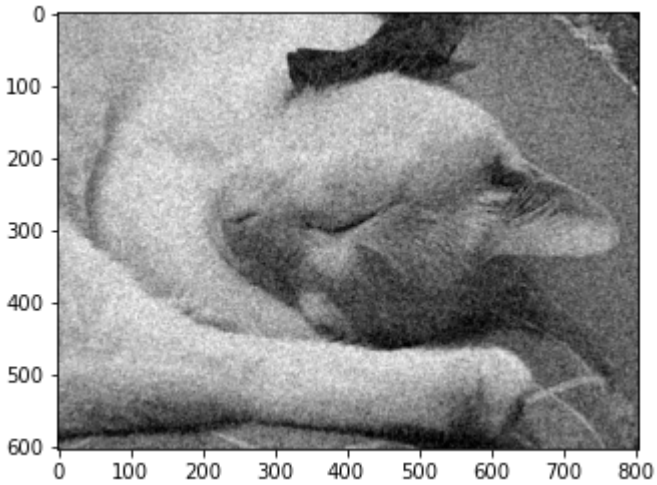
Out[27]: True

In [28]:
```
img_4_spnoise = sp_noise(img_4_gray)
show_image(img_4_spnoise)
cv2.imwrite('result_images/IMG-4-spnoise.jpg', img_4_spnoise)
```



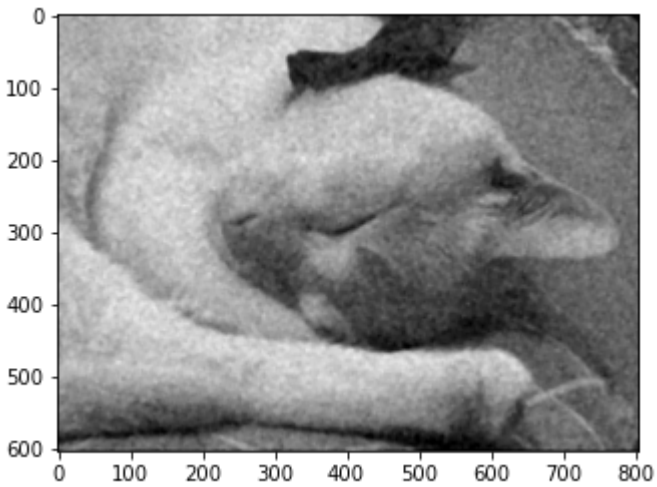
Out[28]: True

In [29]:
```
img_4_filtered = median_filter(img_4_gnoise, 3)
show_image(img_4_filtered)
cv2.imwrite('result_images/IMG-4-g-medain-3.jpg', img_4_filtered)
```



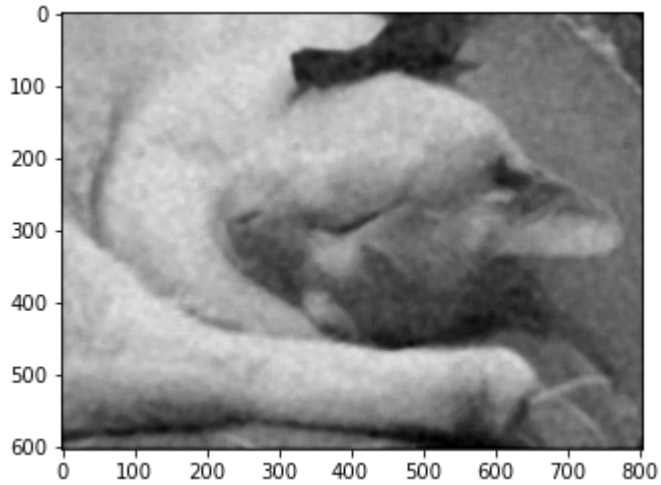
Out[29]: True

In [30]:
```
img_4_filtered = median_filter(img_4_gnoise, 7)
show_image(img_4_filtered)
cv2.imwrite('result_images/IMG-4-g-medain-7.jpg', img_4_filtered)
```



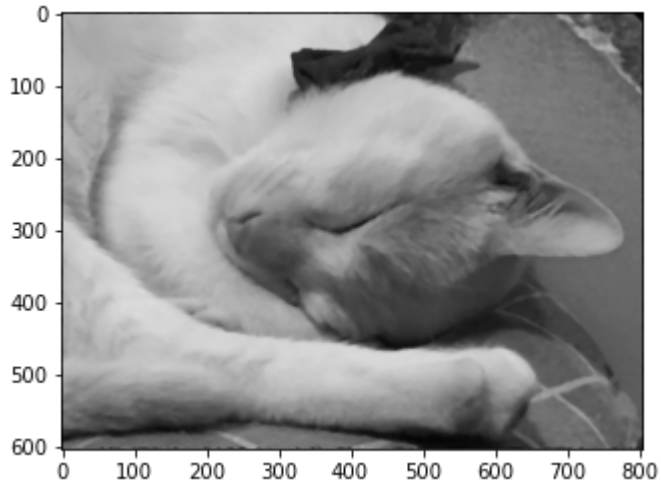
Out[30]: True

In [31]:
```
img_4_filtered = median_filter(img_4_gnoise, 11)
show_image(img_4_filtered)
cv2.imwrite('result_images/IMG-4-g-medain-11.jpg', img_4_filtered)
```



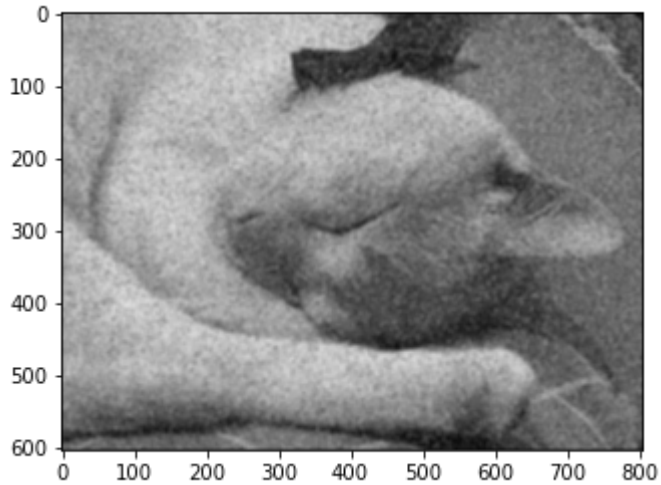
Out[31]: True

In [32]:
```
img_4_filtered = median_filter(img_4_spnoise, 7)
show_image(img_4_filtered)
cv2.imwrite('result_images/IMG-4-sp-medain-7.jpg', img_4_filtered)
```



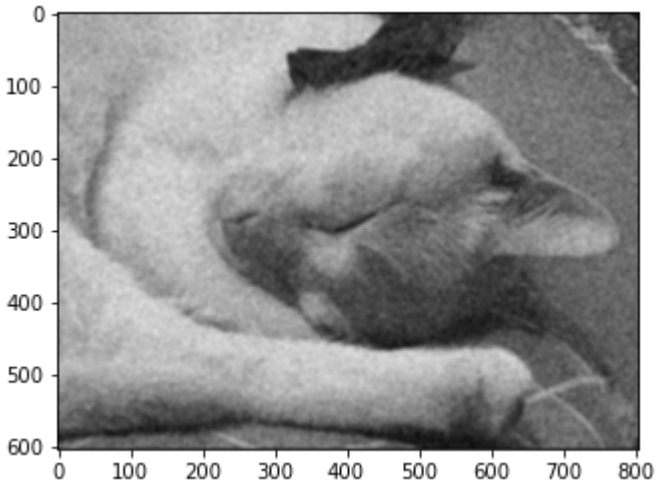
Out[32]: True

In [53]:
```
img_4_filtered = gaussian_filter(img_4_spnoise, 7, 5)
show_image(img_4_filtered)
cv2.imwrite('result_images/IMG-4-sp-g-7-5.jpg', img_4_filtered)
```



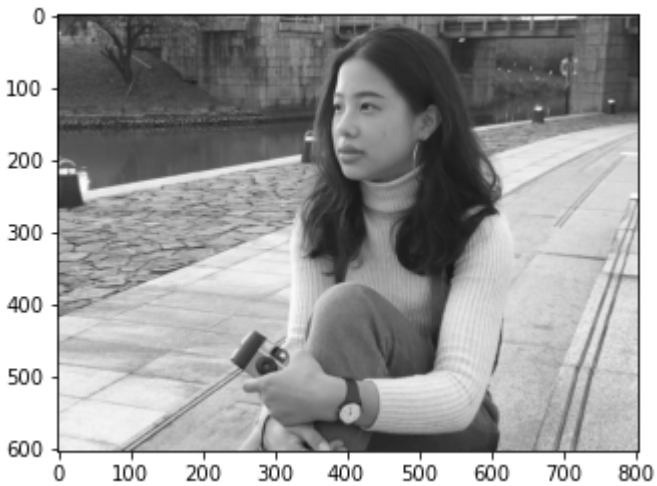
Out[53]: True

In [52]:
```
img_4_filtered = gaussian_filter(img_4_gnoise, 7, 5)
show_image(img_4_filtered)
cv2.imwrite('result_images/IMG-4-g-g-7-5.jpg', img_4_filtered)
```



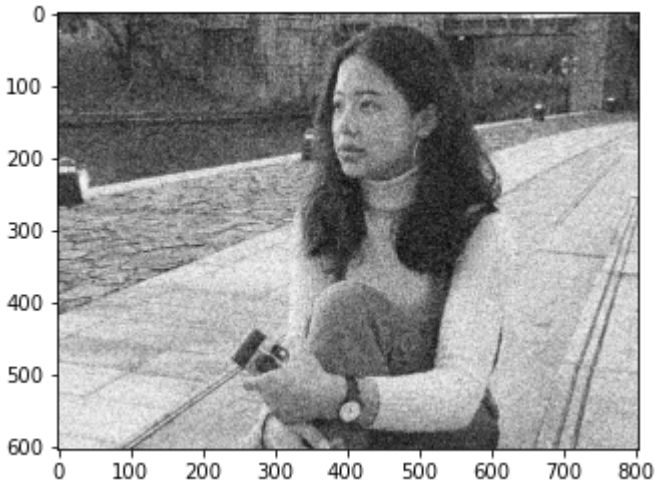
Out[52]: True

In [35]:
```
img_5_small = cv2.resize(img_5,
                         (int(img_5.shape[1]/5), int(img_5.shape[0]/5)),
                         interpolation=cv2.INTER_AREA)
img_5_gray = to_grayscale(img_5_small)
show_image(img_5_gray)
cv2.imwrite('result_images/IMG-5-gray.jpg', img_5_gray)
```



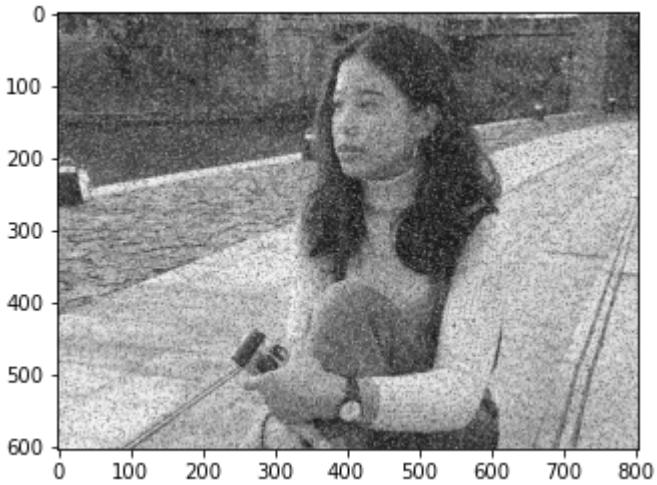
Out[35]: True

In [36]:
```
img_5_gnoise = gaussian_noise(img_5_gray, 0, 0.2)
show_image(img_5_gnoise)
cv2.imwrite('result_images/IMG-5-gnoise.jpg', img_5_gnoise)
```



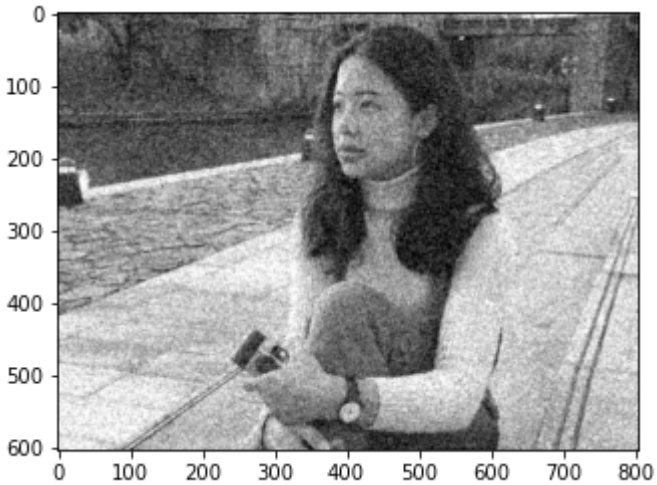
Out[36]: True

In [37]:
```
img_5_spnoise = sp_noise(img_5_gray)
show_image(img_5_spnoise)
cv2.imwrite('result_images/IMG-5-spnoise.jpg', img_5_spnoise)
```



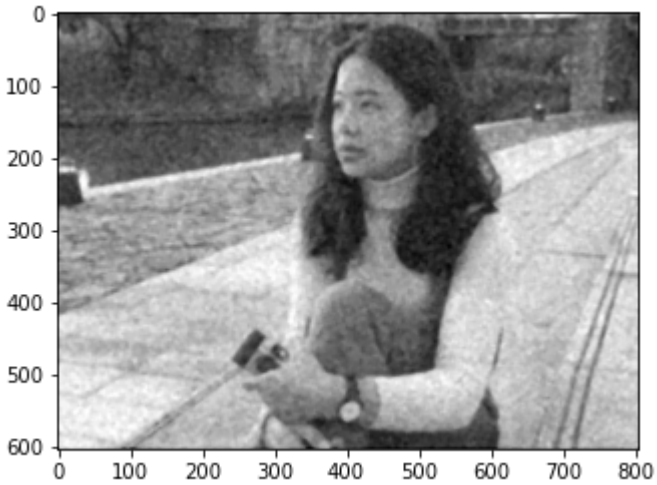
Out[37]: True

In [38]:
```
img_5_filtered = median_filter(img_5_gnoise, 3)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-median-3.jpg', img_5_filtered)
```



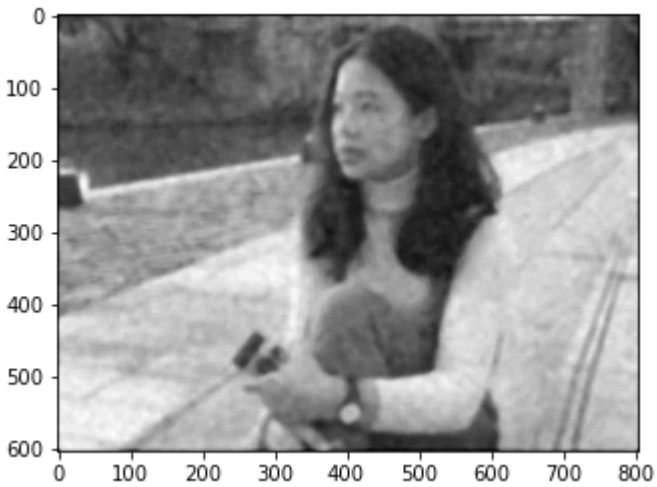
Out[38]: True

In [39]:
```
img_5_filtered = median_filter(img_5_gnoise, 7)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-median-7.jpg', img_5_filtered)
```



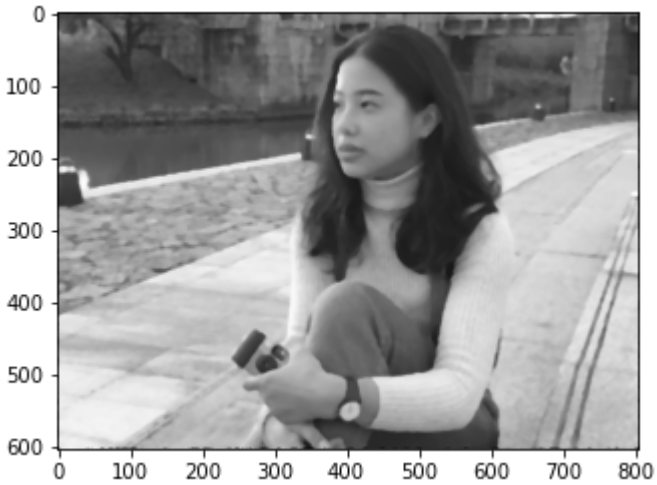
Out[39]: True

In [40]:
```
img_5_filtered = median_filter(img_5_gnoise, 11)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-median-11.jpg', img_5_filtered)
```



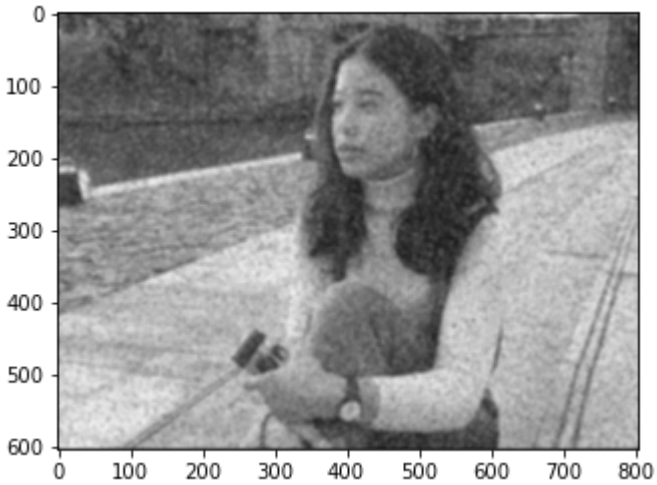
Out[40]: True

In [41]:
```
img_5_filtered = median_filter(img_5_spnoise, 7)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-sp-median-7.jpg', img_5_filtered)
```



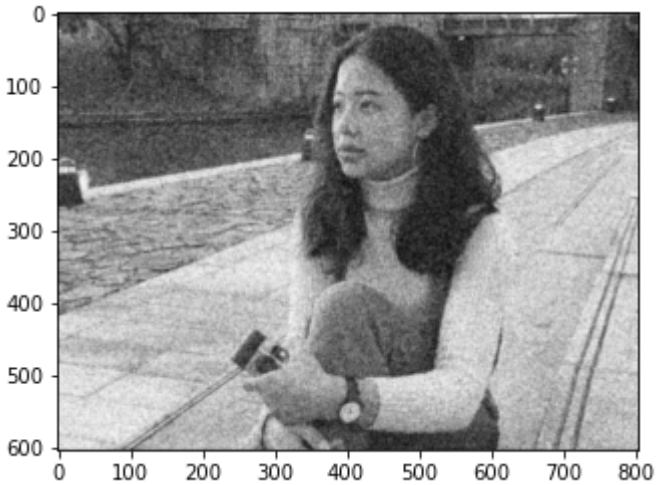
Out[41]: True

In [54]:
```
img_5_filtered = gaussian_filter(img_5_spnoise, 7, 5)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-sp-g-7-5.jpg', img_5_filtered)
```



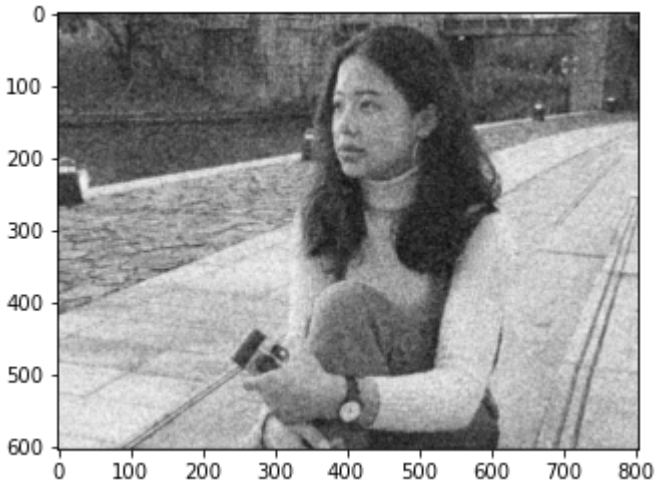
Out[54]: True

In [43]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 7, 1)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-7-1.jpg', img_5_filtered)
```



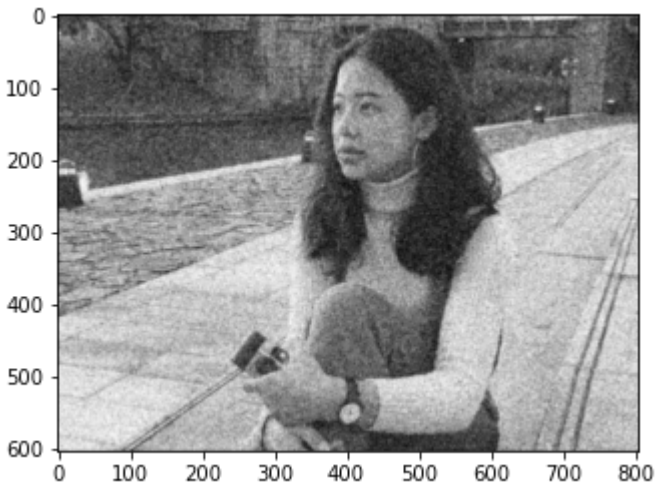
Out[43]: True

In [44]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 11, 1)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-11-1.jpg', img_5_filtered)
```



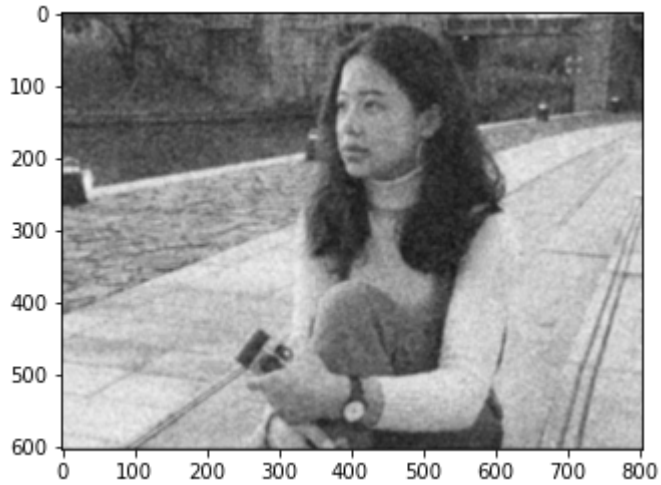
Out[44]: True

In [45]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 13, 1)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-13-1.jpg', img_5_filtered)
```



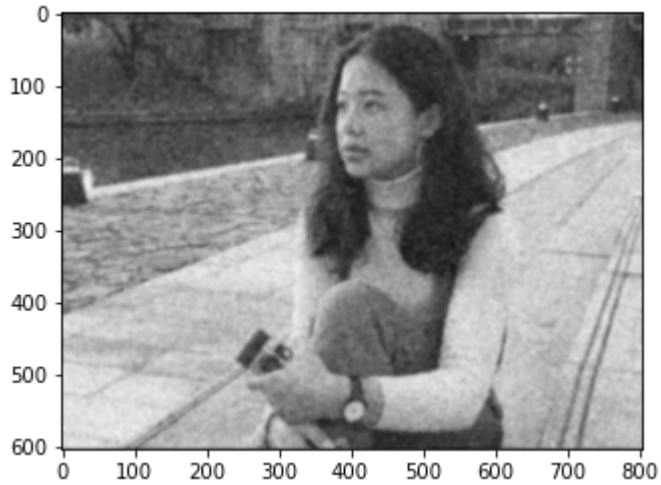
Out[45]: True

In [46]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 7, 2)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-7-2.jpg', img_5_filtered)
```



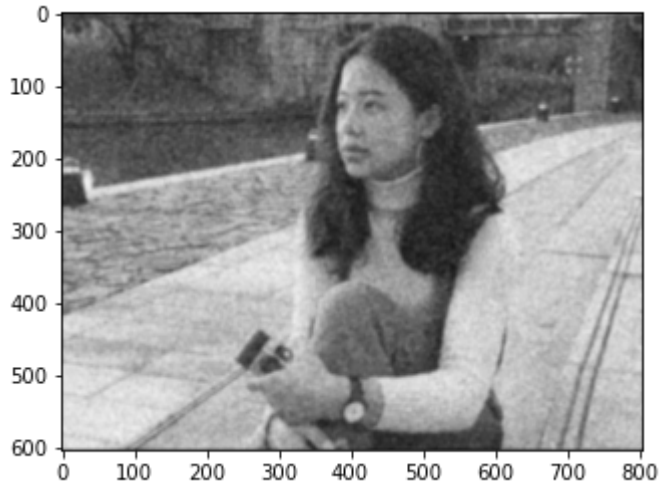
Out[46]: True

In [47]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 11, 2)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-11-2.jpg', img_5_filtered)
```



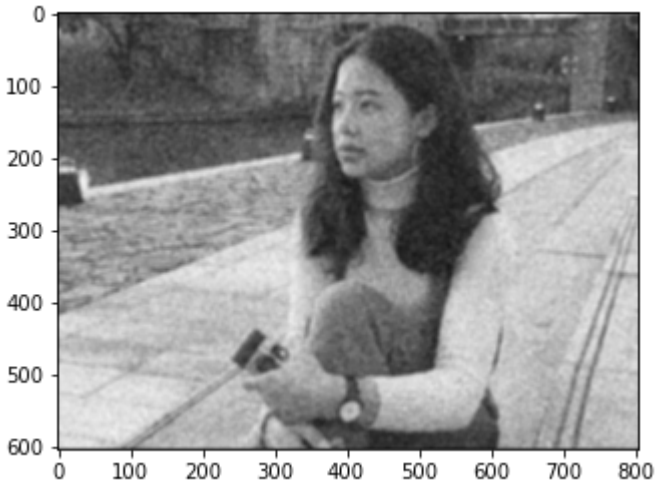
Out[47]: True

In [48]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 13, 2)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-13-2.jpg', img_5_filtered)
```



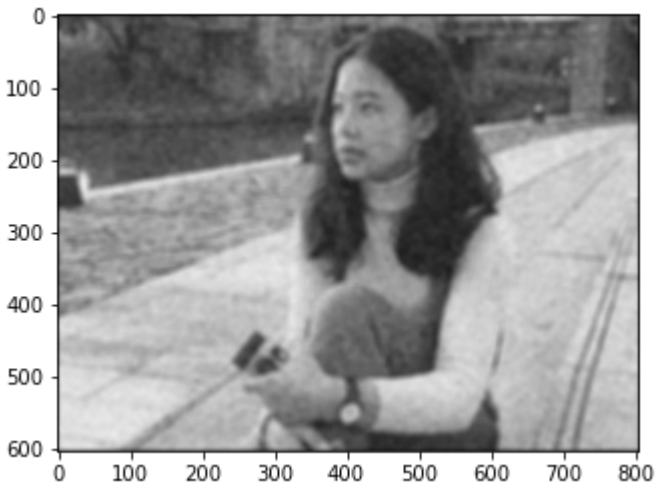
Out[48]: True

In [49]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 7, 5)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-7-5.jpg', img_5_filtered)
```



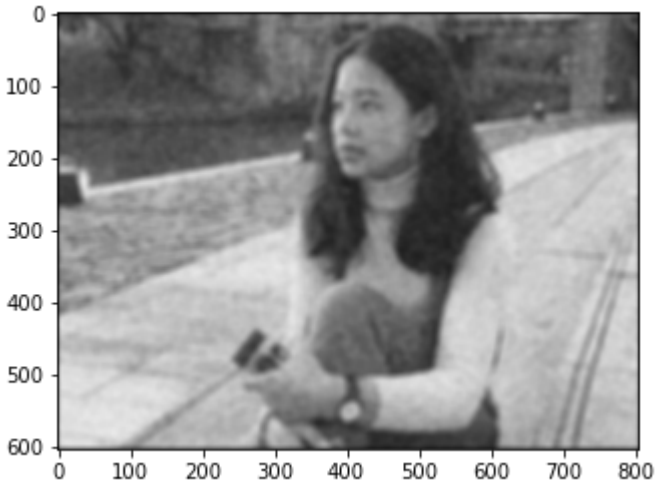
Out[49]: True

In [50]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 11, 5)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-11-5.jpg', img_5_filtered)
```



Out[50]: True

In [51]:
```
img_5_filtered = gaussian_filter(img_5_gnoise, 13, 5)
show_image(img_5_filtered)
cv2.imwrite('result_images/IMG-5-g-g-13-5.jpg', img_5_filtered)
```



Out[51]: True

In [ ]: