NTNU 影像處理 HW9

廖家緯

2020.5.13

- **Outline**:

  1. Create an image $g(x, y)$ whose pixels all have the same gray value of 100. Show the image $g(x, y)$.
  2. Generate Gaussian noise $n(x, y)$, with $\mu = 0, \sigma^2 = 15$, using methods 1 and 2. Show the noisy image $f(x, y) = g(x, y) + n(x, y)$.
  3. Display the histogram $h(i)$ of $f(x, y)$.
  4. Comment on your results.

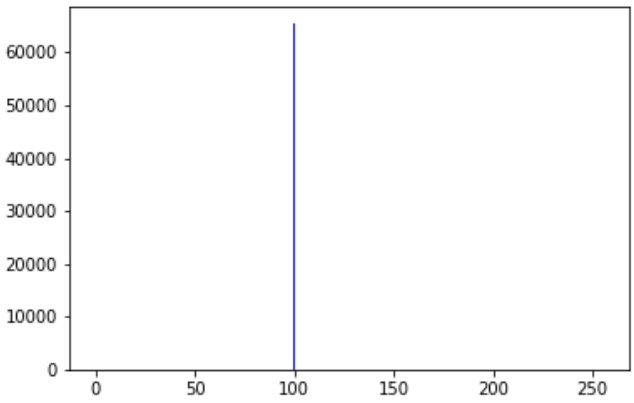
- **Code(Python):**

```python
# coding: utf-8
import numpy as np
import matplotlib.pyplot as plt
import cv2
import random
import math

n, m = 256, 256
g = (np.ones((n, m))*100).astype('uint8')

mu = 0
sigma = 15**(1/2)

#method1
f = (np.zeros((n, m))).astype('uint8')
for i in range(n):
    for j in range(m):
        if j%2 == 0:
            r, phi = random.random(), random.random()
            z1 = sigma*math.cos(2*math.pi*phi)*((-2)*math.log(r))**(1/2)
            z2 = sigma*math.sin(2*math.pi*phi)*((-2)*math.log(r))**(1/2)
            f[i, j] = g[i, j] + z1
            f[i, j+1] = g[i, j+1] + z2

```

```
f1 = (np.zeros((n, m))).astype('uint8')

for i in range(n):
    for j in range(m):
        if f[i, j] == 0:
            f1[i, j] == 0
        elif f[i, j] > 255:
            f1[i, j] = 255
        else:
            f1[i, j] = f[i, j]

#method2
f2 = (np.zeros((n, m))).astype('uint8')

for i in range(n):
    for j in range(m):
        s = np.random.normal(mu, sigma)
        f2[i, j] = g[i, j] + s
```

- **Input image:**

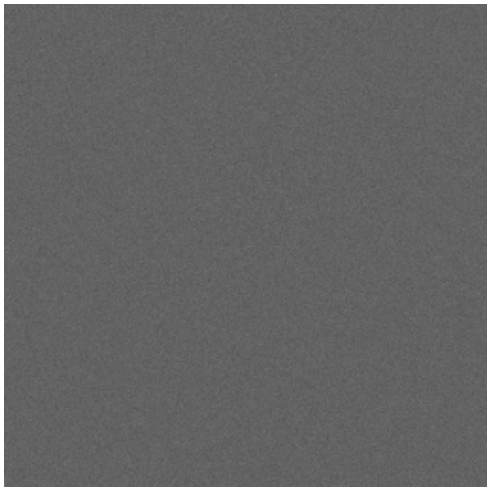
Input image $g(x, y)$



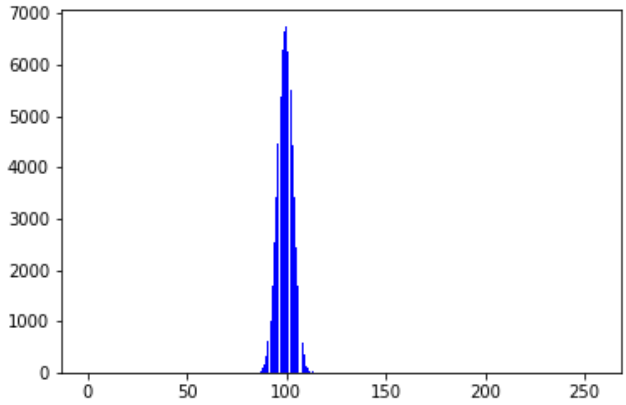
Histogram of $g(x, y)$

- **Result:**

  (1) Method 1

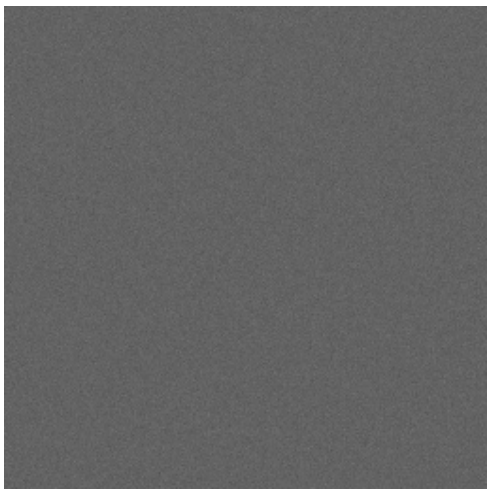


Noisy image $f(x, y)$

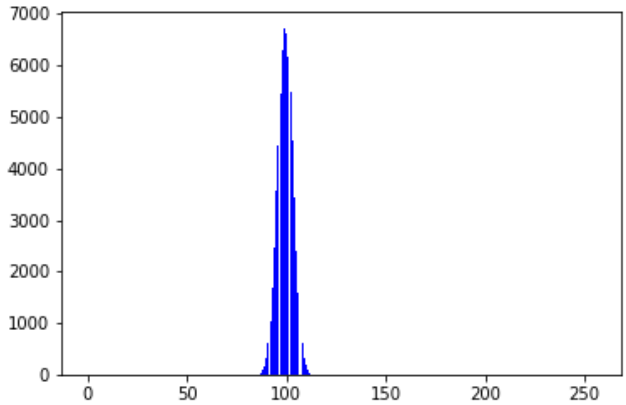


Histogram of $f(x, y)$

  (2) Method 2



Noisy image $f(x, y)$



Histogram of $f(x, y)$

- **Experience:**

  這次作業花了點時間想 method2 的原理，我使用內建的常態分配 (高斯分配)random 取一個值，機率高的取到的次數多，而機率小的取到的次數小，符合老師說的機率大，對應到的區間大，機率小，對應到的區間小。從結果來看 method1 與 method2 差不多，可能 σ 要調整更大才會有明顯差異。