1. Method

Part1: 首先，因為原始圖片已經是灰階圖片，所以省去轉灰階的步驟。接著計算像素值的累積分布函數(cdf_1)，按照公式:

$$\text{color value} = \text{round}(255 \text{ x } \text{cdf1}[\text{color value}])$$

將 color_value 存回原始影像對應位置的三個圖像通道，即可完成 Histogram Equalization。

Part2: 仿照在 Part1 中的方法，算出 Q1 和 Q2 兩張圖片各自的累積分布函數(cdf_1, cdf_2)，並實作 inverse_map:

```
48    inverse_map = np.full((256), -1)
49
50    for i in range(255, -1, -1):
51        inverse_map[round(255 * cdf_2[i])] = i
```

若 inverse_map 返回值存在，則將其存回原始影像對應位置的三個圖像通道；否則存回經轉換後與 color_value 最接近且數值最小的 k。

```
55  color_value = round(255 * cdf_1[img_1[i, j, 0]])
56  if inverse_map[color_value] == -1:
57      k = 1
58      while True:
59          if color_value - k >= 0 and inverse_map[color_value - k] != -1:
60              img_1[i, j, 0], img_1[i, j, 1], img_1[i, j,
61                                              2] = inverse_map[color_value - k], inverse_map[color_value - k], inverse_map[color_value
62              break
63
64          elif color_value + k <= 255 and inverse_map[color_value + k] != -1:
65              img_1[i, j, 0], img_1[i, j, 1], img_1[i, j,
66                                              2] = inverse_map[color_value + k], inverse_map[color_value + k], inverse_map[color_value
67              break
68
69          k += 1
70  else:
71      img_1[i, j, 0], img_1[i, j, 1], img_1[i, j,
72                                      2] = inverse_map[color_value], inverse_map[color_value], inverse_map[color_value]
```

Part3: 先對原圖做邊界填補(numpy.pad(img_3, ((2, 2), (2, 2)))，接著照

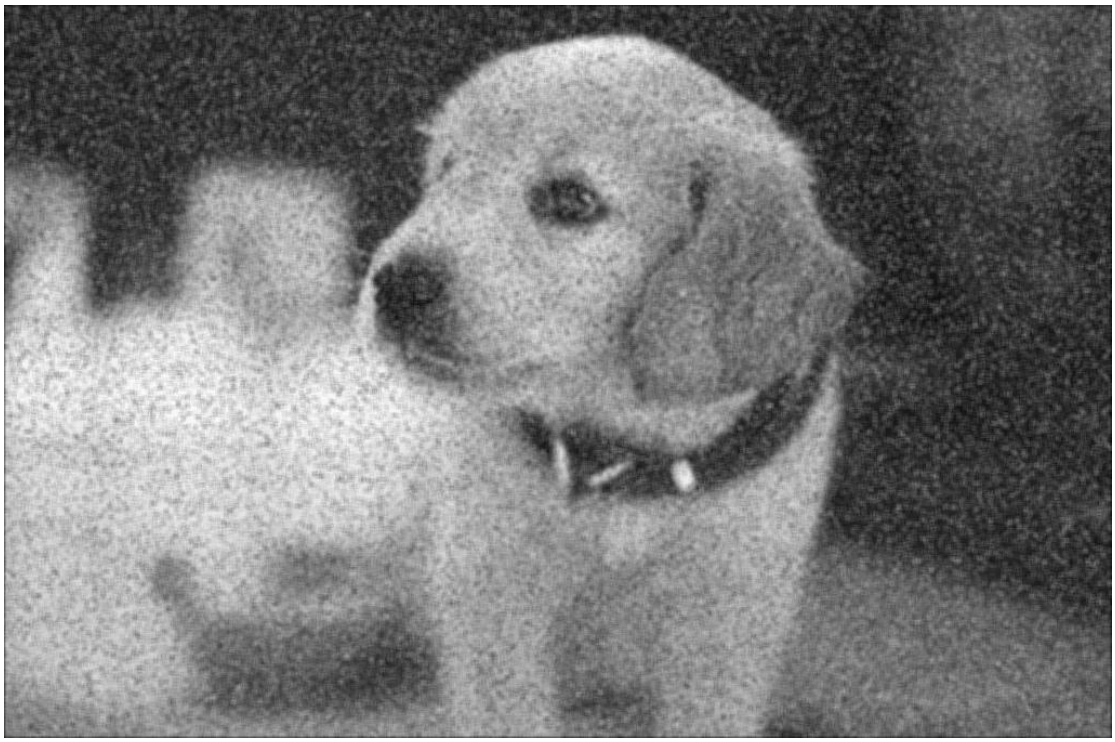
公式得到矩陣並掃過整張圖片計算結果。

2. Result

Histogram Equalization:

Histogram Specification:

Gaussian Filter (K=1, size=5x5, σ=25):



3. Feedback

希望下次作業能給結果圖片供對照參考，而像 Part 3 這種轉換前後差別不太大的圖片，提供輸出結果的數值(如:整張圖前幾個 pixel 和後幾個 pixel 大約的數值)，我們會比較好評估自己的作業是否做對!