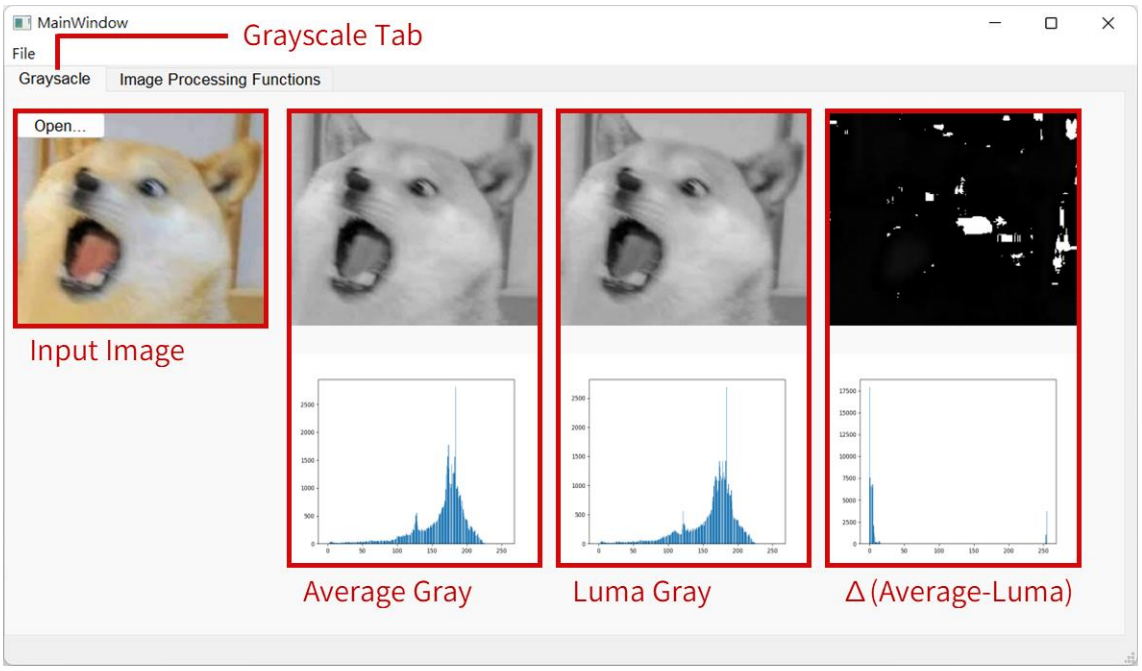
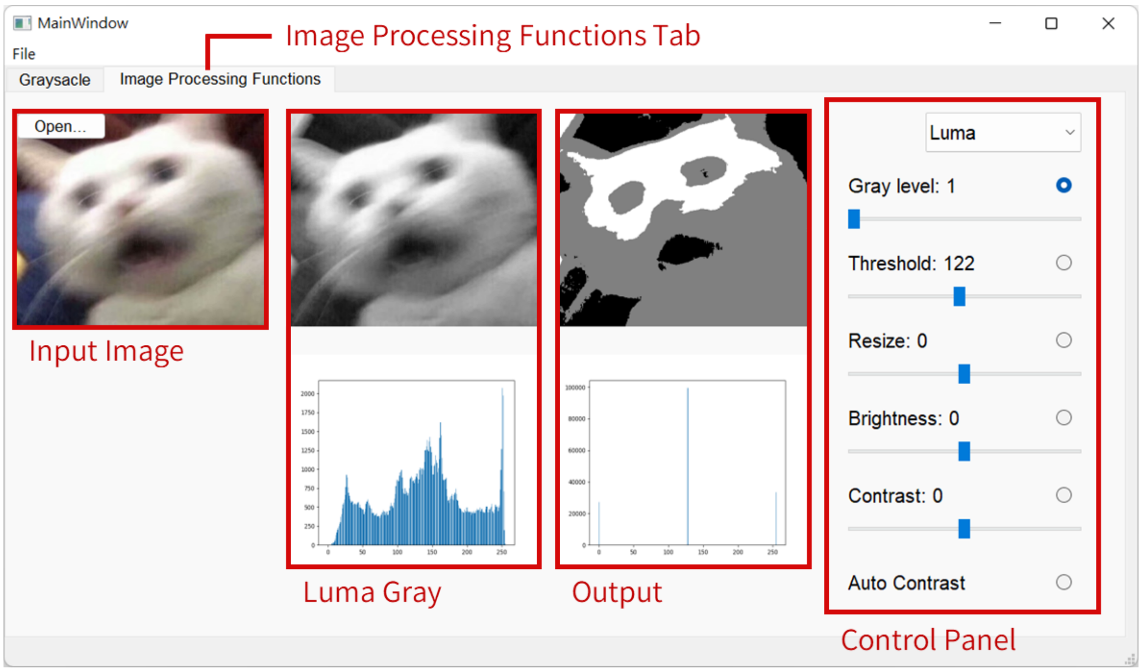
Principles and Applications of Digital Image Processing

Homework 2

GUI





Luma
Gray level: 1 [✓]
Threshold: 122
Resize: 0
Brightness: 0
Contrast: 0
Auto Contrast

圖一、介面使用說明。(a) Grayscale images，(b) Processing Images。

Part 1: (30%)

2.12

$$0 \le i(x,y) = 255 \times e^{-[(x-x_0)^2+(y-y_0)^2]} \le 255$$

$$2^k \cdot 8 = 255 + 1 = 2^8, \quad k = 5$$

2.16 # from m-path to 4-path,

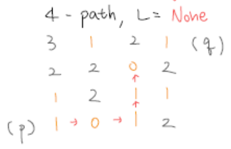
```
If   q == N4(p)
     continue,
else: # p to q, one-pixel-thick
     δx = x_q - x_p
     δy = y_q - y_p

     path[0] = (x_p + δx, y_p)
     path[1] = (x_p, y_p + δy)
```

2.18

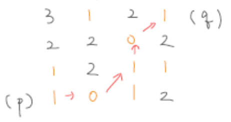
(a)

$N_4(q) \in V'$

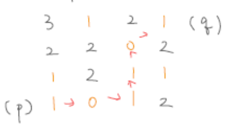
4-path, L= None



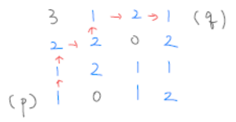
8-path, L= 4



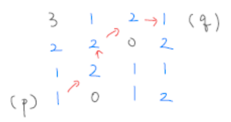
m-path, L= 5



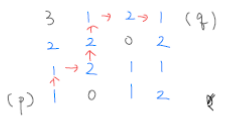
(b). 4-path, L= 6



8-path, L= 4



m-path, L= 6



2.37

(a.) scaling.

$$A^{-1} = \begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{c_x} & 0 & 0 \\ 0 & \frac{1}{c_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b.) translation

$$A^{-1} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$x' = x + t_x \qquad x = x' - t_x$

$y' = y + t_y \qquad y = y' - t_y$

(c). Shearing.

$$A_v^{-1} = \begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$x' = x + s_v y, \; y' = y$

$x = x' - s_v y,$

$y' = s_h x + y$

$, \; y = y' - s_h x$

$$A_h^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(d) rotation

$$A^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} -\cos\theta & -\sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\theta' = -\theta$

(e) translation / rotation

$$A^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\cos\theta & -\sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\cos\theta & -\sin\theta & -t_x \\ \sin\theta & -\cos\theta & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

3.12

$p_r(r) = -2r + 2$

$p_z(z) = 2z$

$\int_0^r -2w+2 \quad -w^2+2w \quad -r^2+2r$

$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$

$= (L-1)(-r^2+2r)$

v^2

$G(z) = (L-1)\int_0^z p_z(v)\,dv = s$

$= (L-1) \cdot z^2 = (L-1)(-r^2+2r)$

$z = \sqrt{-r^2+2r}$

3.18

a.)

$$f \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad w \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$(w \bigstar f)(2,3) = 1 \cdot 4 + 1 \cdot 2 = 6$

$$w \bigstar f = \begin{bmatrix} 0&0&0&0&0&0&0 \\ 0&0&1&2&1&0&0 \\ 0&0&3&6&3&0&0 \\ 0&0&4&8&4&0&0 \\ 0&0&3&6&3&0&0 \\ 0&0&1&2&1&0&0 \\ 0&0&0&0&0&0&0 \end{bmatrix}$$

b.)

Rotated w equals to w.

$$(w \bigstar f)(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

(c).

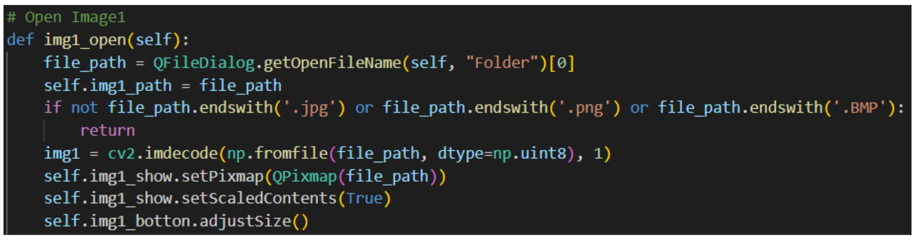
$$(w \star f)(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$(w \star f)(2,3) = 1 \cdot 4 + 1 \cdot 2 = 6$

$$w \star f = \begin{bmatrix} 0&0&0&0&0&0&0 \\ 0&0&1&2&1&0&0 \\ 0&0&3&6&3&0&0 \\ 0&0&4&8&4&0&0 \\ 0&0&3&6&3&0&0 \\ 0&0&1&2&1&0&0 \\ 0&0&0&0&0&0&0 \end{bmatrix}$$

Part 2: (70%) Image File Reading, Display and Basic Processing

1. Read a color BMP or JPEG image file and display it on the screen. You may use the functions provided by Qt, OpenCV, or MATLAB to read and display an image file. (10%)



```python
# Open Image1
def img1_open(self):
    file_path = QFileDialog.getOpenFileName(self, "Folder")[0]
    self.img1_path = file_path
    if not file_path.endswith('.jpg') or file_path.endswith('.png') or file_path.endswith('.BMP'):
        return
    img1 = cv2.imdecode(np.fromfile(file_path, dtype=np.uint8), 1)
    self.img1_show.setPixmap(QPixmap(file_path))
    self.img1_show.setScaledContents(True)
    self.img1_botton.adjustSize()
```

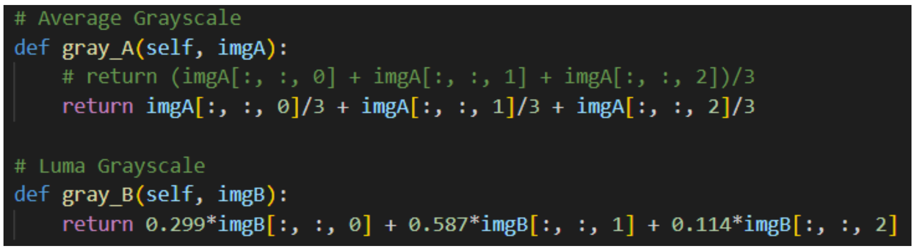
2. Convert a color image into a grayscale image using the following equations:

   A. GRAY = (R+G+B)/3.0

   B. GRAY = 0.299*R + 0.587*G + 0.114*B

   Compare the grayscale images obtained from the above equations. One way to compare the difference between two images is by image subtraction (10%)

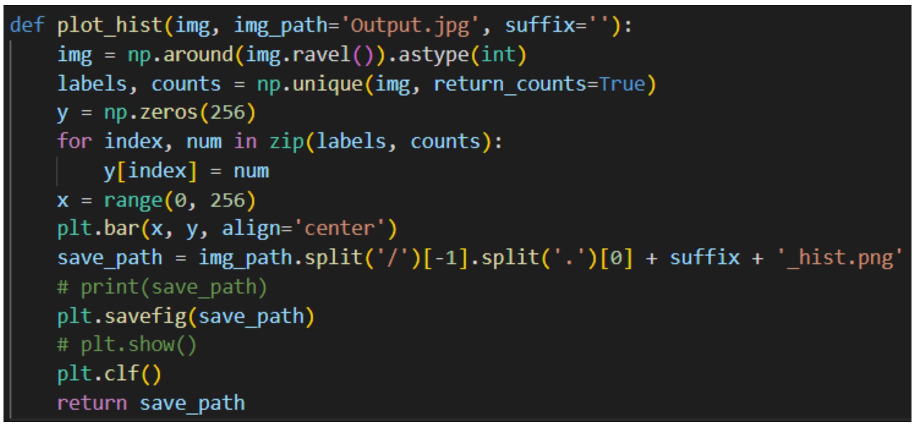
   實作 Average Gray 和 Luma Gray，圖一之最右邊小圖示兩種灰階影像之差異，結果可見圖一(a)。觀看結果圖可見透過 Luma Gray 轉換的視覺效果較為飽滿。



```python
# Average Grayscale
def gray_A(self, imgA):
    # return (imgA[:, :, 0] + imgA[:, :, 1] + imgA[:, :, 2])/3
    return imgA[:, :, 0]/3 + imgA[:, :, 1]/3 + imgA[:, :, 2]/3

# Luma Grayscale
def gray_B(self, imgB):
    return 0.299*imgB[:, :, 0] + 0.587*imgB[:, :, 1] + 0.114*imgB[:, :, 2]
```

3. Determine and display the histogram of a grayscale image. (10%)

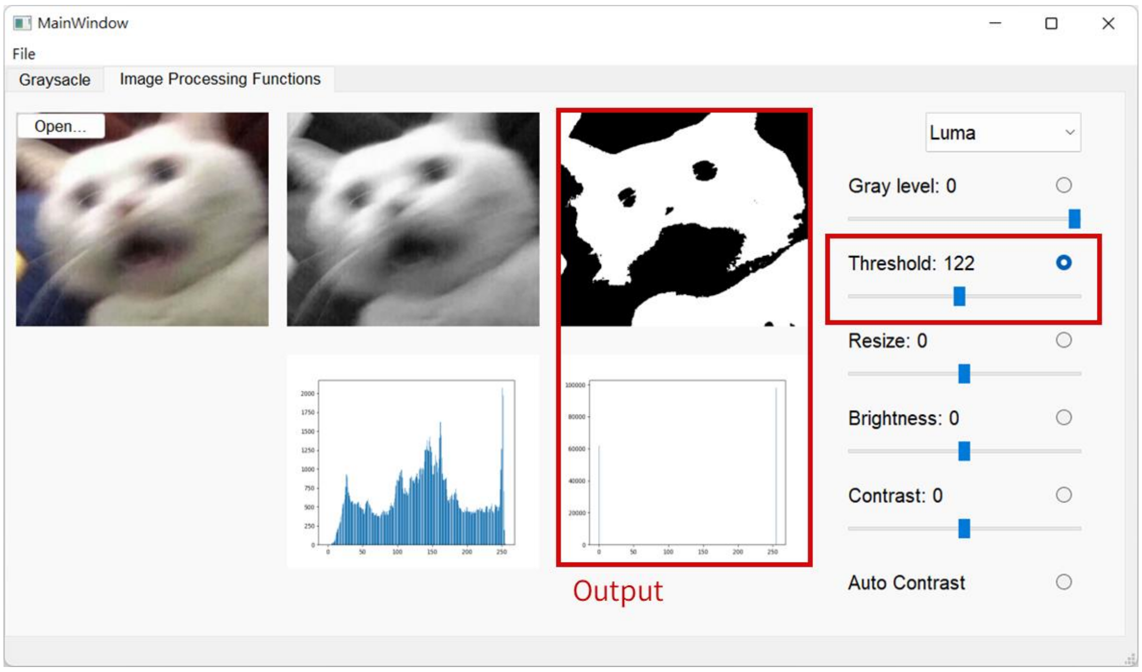
   透過 plt.bar 繪圖，結果可見圖一(a)。



```python
def plot_hist(img, img_path='Output.jpg', suffix=''):
    img = np.around(img.ravel()).astype(int)
    labels, counts = np.unique(img, return_counts=True)
    y = np.zeros(256)
    for index, num in zip(labels, counts):
        y[index] = num
    x = range(0, 256)
    plt.bar(x, y, align='center')
    save_path = img_path.split('/')[-1].split('.')[0] + suffix + '_hist.png'
    # print(save_path)
    plt.savefig(save_path)
    # plt.show()
    plt.clf()
    return save_path
```

4. Implement a manual threshold function to convert a grayscale image into a binary image. (10%)
圖二是二值化 Luma 灰階影像的輸出結果。將值設為 122，可見白貓的白毛區塊明顯地從背景凸顯出來，影像的數值分布只剩下 0 與 255。

```python
def thresh(self):
    threshold = self.thresh_slider.value()
    self.output_arr = self.img2_switch[self.comboBox.currentIndex()]
    self.output_arr = np.where(self.output_arr > threshold, 255, 0)
    self.show_ouput()
```



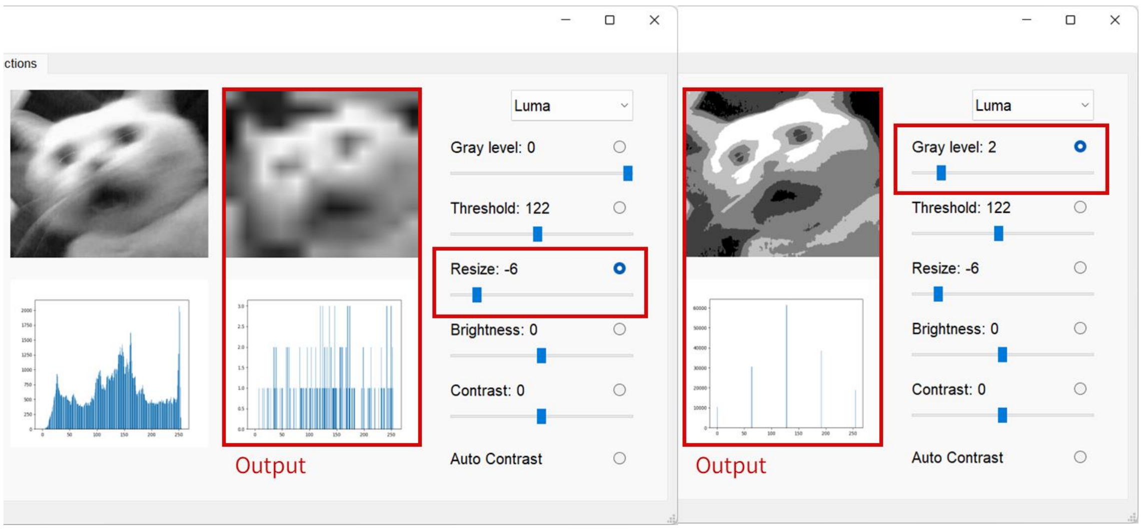
Luma
Gray level: 0
[✓ Threshold: 122]
Resize: 0
Brightness: 0
Contrast: 0
Auto Contrast

圖二、二值化輸出結果。

5. Implement a function to adjust the spatial resolution (enlarge or shrink) and grayscale levels of an image. Use interpolation on enlarging the image. (10%)
使用 Bilinear Scaling Algorithm 實作 resize function，圖三(a)是縮小 2^(6/1.2)倍後的影像，可見影像倍縮小後流失許多顏色資訊，影像變得模糊且像素感強烈。圖三(b)是將 gray level 設為 2^2+1 的灰階輸出影像，影像只剩下五種顏色，白貓變得模糊，顏色失去深度。

```python
def resize_img(self):
    img = self.img2_switch[self.comboBox.currentIndex()]
    scale = pow(2, self.resize_slider.value()/1.2)
    row_num = round(img.shape[0]*scale)
    col_num = round(img.shape[1]*scale)
    self.output_arr = np.zeros((row_num, col_num))
    for i in range(row_num):
        for j in range(col_num):
            rf = i/scale
            cf = j/scale
            intr = int(rf)
            intc = int(cf)
            delr = rf - intr
            delc = cf - intc
            self.output_arr[i, j] = img[intr-1, intc-1]*(1-delr)*(1-delc) +\
                img[intr, intc-1]*delr*(1-delc)+img[intr-1, intc]*(1-delr)*delc +\
                img[intr, intc]*delr*delc
    self.show_ouput()
```

```python
def glevel(self):
    gray_level = self.glevel_slider.value()
    self.output_arr = self.img2_switch[self.comboBox.currentIndex()]
    self.output_arr = np.round(
        self.output_arr/pow(2, 8-gray_level))*pow(2, 8-gray_level)
    self.show_ouput()
```


ctions

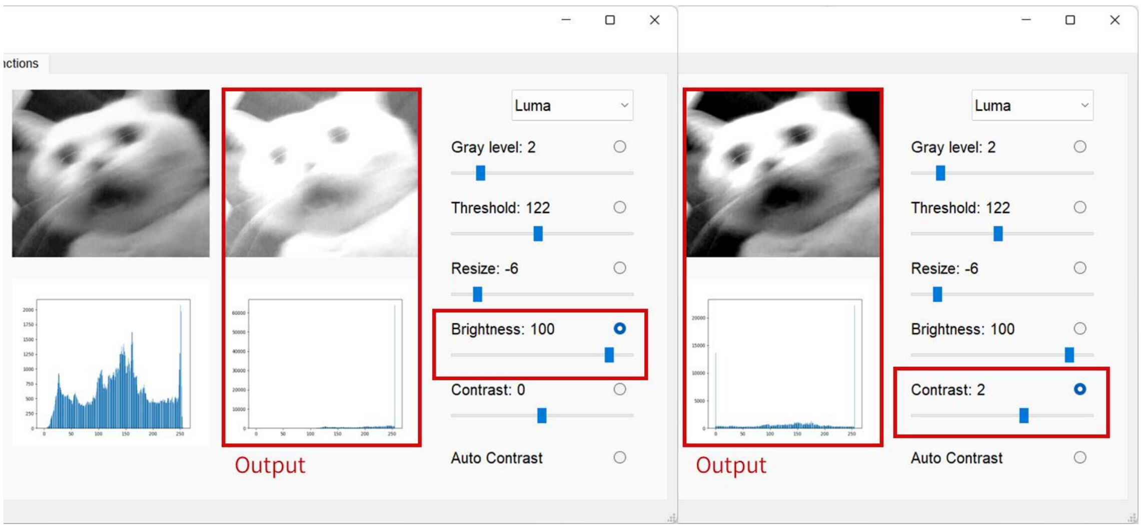
Luma
Gray level: 0
Threshold: 122
Resize: -6
Brightness: 0
Contrast: 0
Auto Contrast
Output

Luma
Gray level: 2
Threshold: 122
Resize: -6
Brightness: 0
Contrast: 0
Auto Contrast
Output

圖三、(a) Resized Image，(b) Gray Level Image。

6. Implement a function to adjust the brightness and contrast of an image. (10%)
   調整影像亮度時使用單純的加減法，若大於 255 則為 255，小於 0 則為 0。圖四(a)亮度加上 100 後呈顯過曝。影像的對比度則透過調整影像的像素數值分布區間達成。圖四(b)經過對比加強後，白貓變得更有立體感。

```python
def contrast(self):
    c = self.contrast_slider.value()
    self.output_arr = self.img2_switch[self.comboBox.currentIndex()]
    self.output_arr = (self.output_arr-255/2) * pow(2, c/4) + 255/2
    self.show_ouput()

def bright(self):
    brightness = self.bright_slider.value()
    self.output_arr = self.img2_switch[self.comboBox.currentIndex()]
    self.output_arr = self.output_arr + brightness
    self.show_ouput()
```

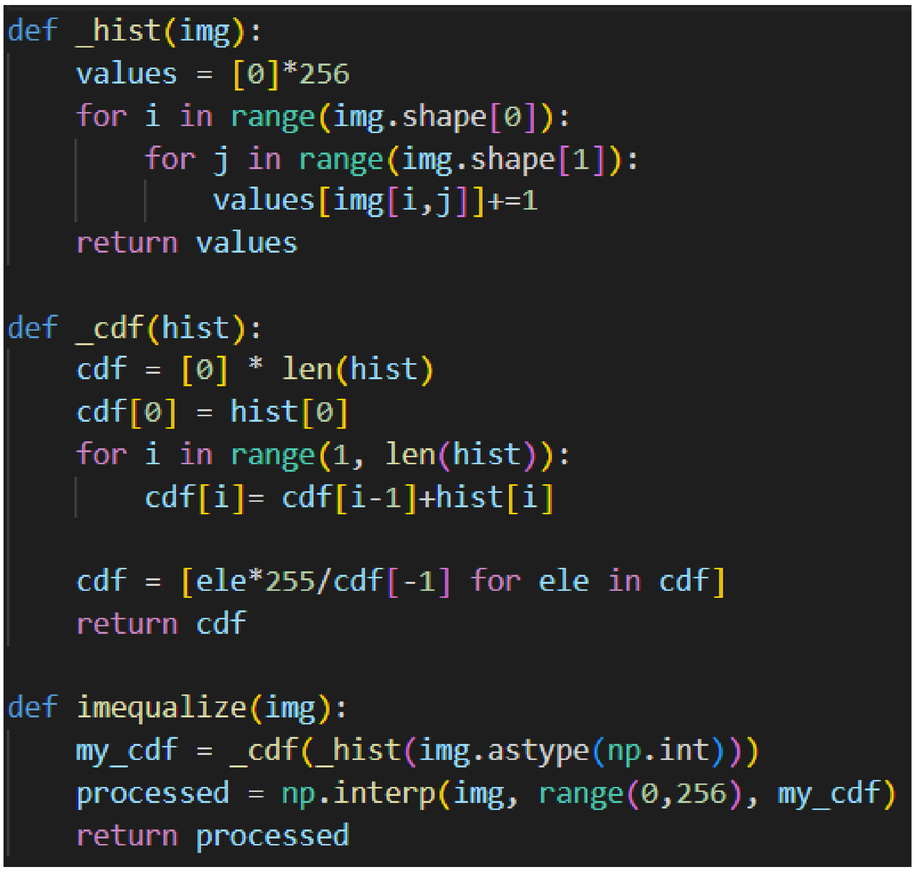

ictions

Luma
Gray level: 2
Threshold: 122
Resize: -6
Brightness: 100
Contrast: 0
Auto Contrast
Output

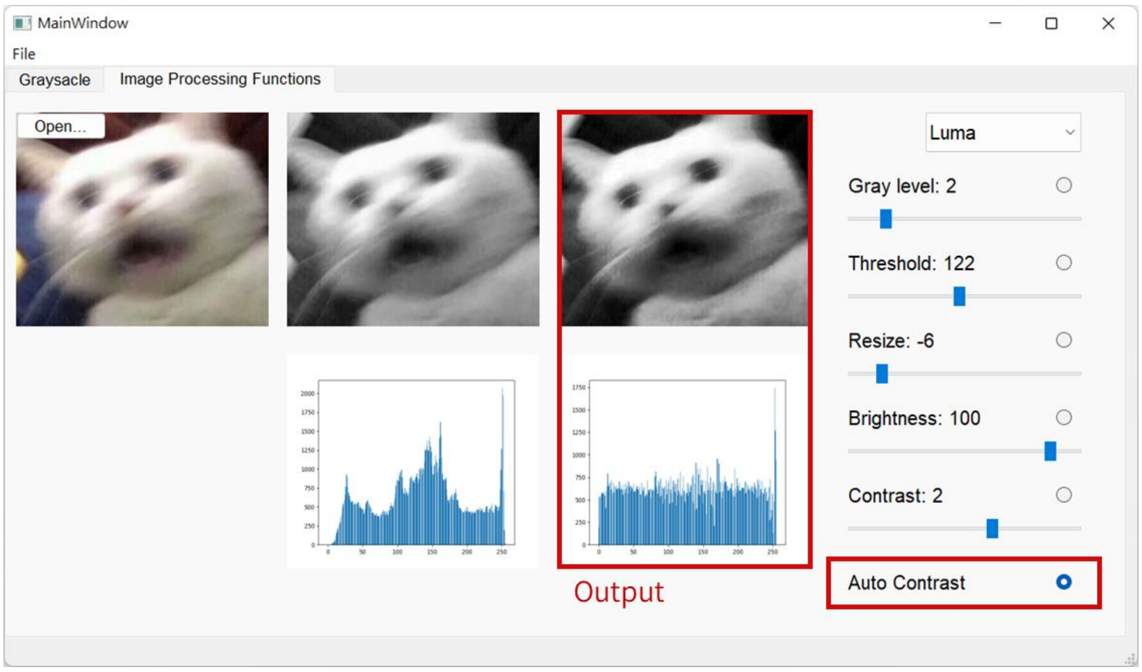
Luma
Gray level: 2
Threshold: 122
Resize: -6
Brightness: 100
Contrast: 2
Auto Contrast
Output

圖四、(a) Brightness Image，(b) Contrast Image。

7. Implement a histogram equalization function for automatic contrast adjustment. (10%)
本題為 GUI 中的 Auto Contrast 功能。可見圖五，輸出之影像的 histogram 分布較為平緩。且不用人為設定影像欲調整的對比度。



```python
def _hist(img):
    values = [0]*256
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            values[img[i,j]]+=1
    return values

def _cdf(hist):
    cdf = [0] * len(hist)
    cdf[0] = hist[0]
    for i in range(1, len(hist)):
        cdf[i]= cdf[i-1]+hist[i]

    cdf = [ele*255/cdf[-1] for ele in cdf]
    return cdf

def imequalize(img):
    my_cdf = _cdf(_hist(img.astype(np.int)))
    processed = np.interp(img, range(0,256), my_cdf)
    return processed
```



Luma

Gray level: 2

Threshold: 122

Resize: -6

Brightness: 100

Contrast: 2

Auto Contrast ●

圖五、經過自動調整對比度之灰階影像。