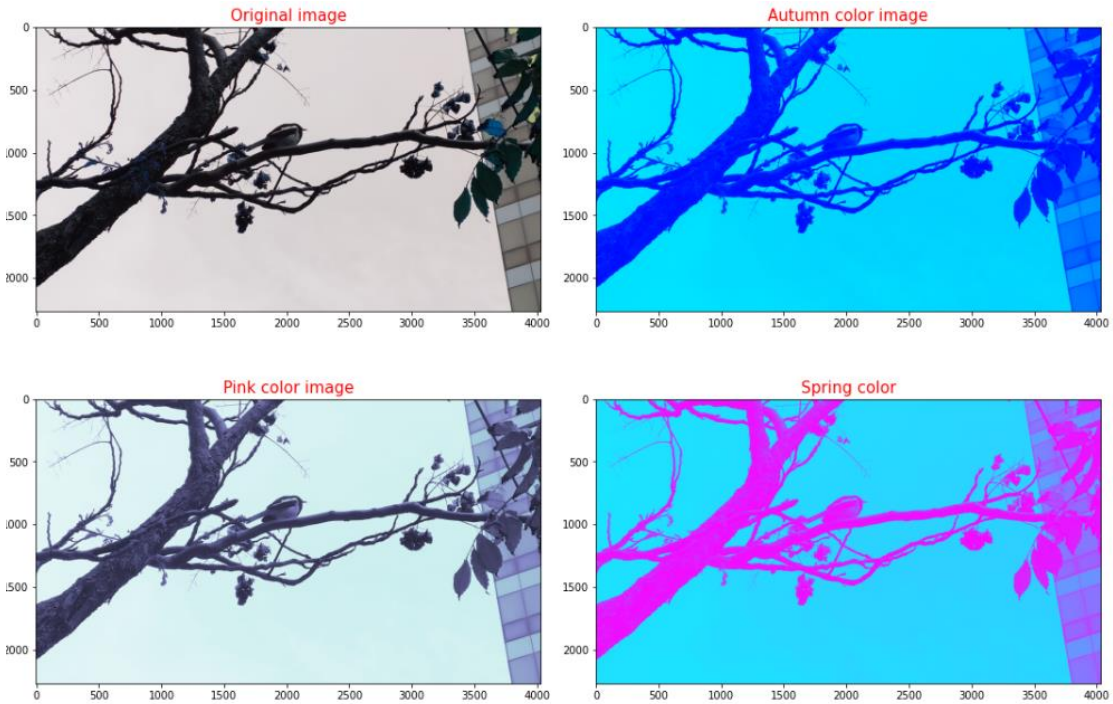
# 實戰二

```python
import numpy as np
import cv2
import math
from matplotlib import pyplot as plt
from google.colab import drive
from google.colab.patches import cv2_imshow

drive.mount('/content/drive')
img = cv2.imread("/content/drive/My Drive/Colab Notebooks/image_processing/week12.bmp", -1)
img1 = cv2.applyColorMap(img,cv2.COLORMAP_AUTUMN)
img2 = cv2.applyColorMap(img,cv2.COLORMAP_PINK)
img3 = cv2.applyColorMap(img,cv2.COLORMAP_SPRING)

imgs = [img, img1, img2, img3]
titles = ['Original image', 'Autumn color image', 'Pink color image', 'Spring color']
plt.figure(figsize = (15, 10))

for i in range(4):
        plt.subplot(2, 2, i + 1)
        plt.title(titles[i], fontsize = 15, color = 'r')
        plt.imshow(imgs[i], cmap = 'gray')

plt.tight_layout()
plt.show()
```

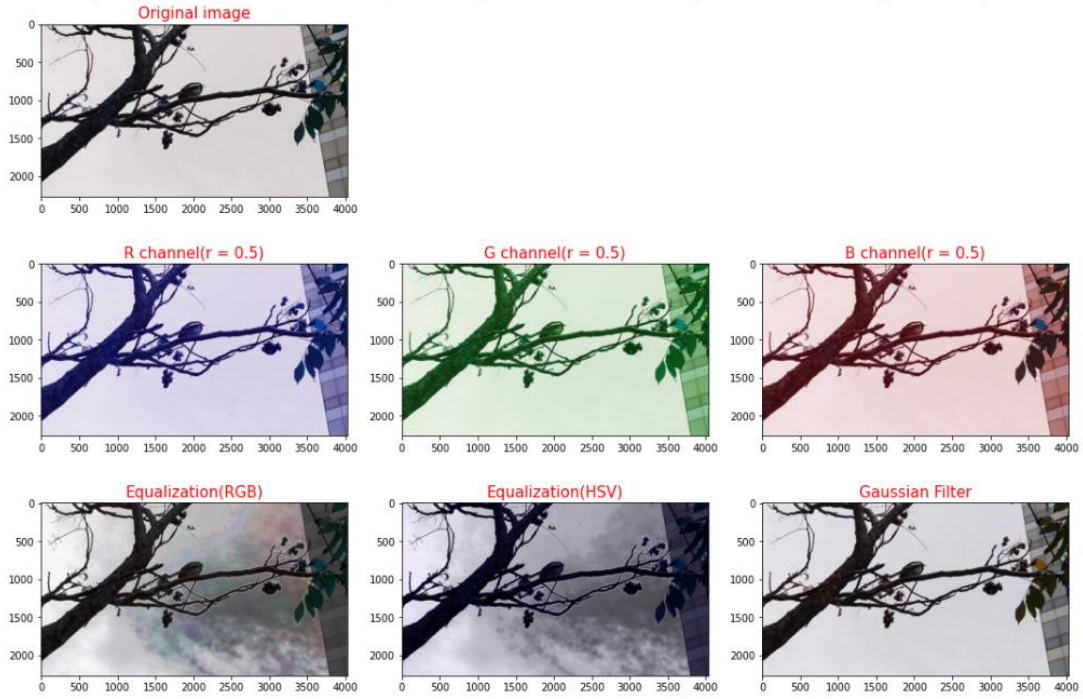



透過上圖可以看出，透過各色彩表所輸出之影像的結果實際顏色差異巨大。

# 實戰三

```python
import numpy as np
import cv2
import math
from matplotlib import pyplot as plt
from google.colab import drive
from google.colab.patches import cv2_imshow

drive.mount('/content/drive')
img = cv2.imread("/content/drive/My Drive/Colab Notebooks/image_processing/week12.bmp", -1)

def RGB_gamma_correction(f, channel, gamma):
        g = f.copy()
        nr, nc = f.shape[:2]
        c = 255.0 / (255.0 ** gamma)
        table = np.zeros(256)

        for i in range(256):
                table[i] = round(i ** gamma * c, 0)

        if channel == 1:
                k = 2
        elif channel == 2:
                k = 1
        else:
                k = 0

        for x in range(nr):
                for y in range(nc):
                        g[x, y, k] = table[f[x, y, k]]

        return g

def RGB_histogram_equalization(f):
        g = f.copy()

        for k in range(3):
                g[:, :, k] = cv2.equalizeHist(f[:, :, k])

        return g

def HSV_histogram_equalization( f ):
        hsv = cv2.cvtColor(f, cv2.COLOR_BGR2HSV)
        hsv[:, :, 2] = cv2.equalizeHist(hsv[:, :, 2])
        g = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

        return g
```

```python
img1 = RGB_gamma_correction(img, 1, 0.5)
img2 = RGB_gamma_correction(img, 2, 0.5)
img3 = RGB_gamma_correction(img, 3, 0.5)
img4 = RGB_histogram_equalization(img1)
img5 = HSV_histogram_equalization(img1)
img6 = cv2.GaussianBlur(cv2.cvtColor(img, cv2.COLOR_BGR2RGB), (5, 5), 0)

imgs = [img, img1, img2, img3, img4, img5, img6]
titles = ['Original image', 'R channel(r = 0.5)', 'G channel(r = 0.5)', 'B channel(r = 0.5)',
          'Equalization(RGB)', 'Equalization(HSV)', 'Gaussian Filter']
plt.figure(figsize = (15, 10))

for i in range(7):
        if(i == 0):
                plt.subplot(3, 3, i + 1)
        else:
                plt.subplot(3, 3, i + 3)
        plt.title(titles[i], fontsize = 15, color = 'r')
        plt.imshow(imgs[i], cmap = 'gray')

plt.tight_layout()
plt.show()
```



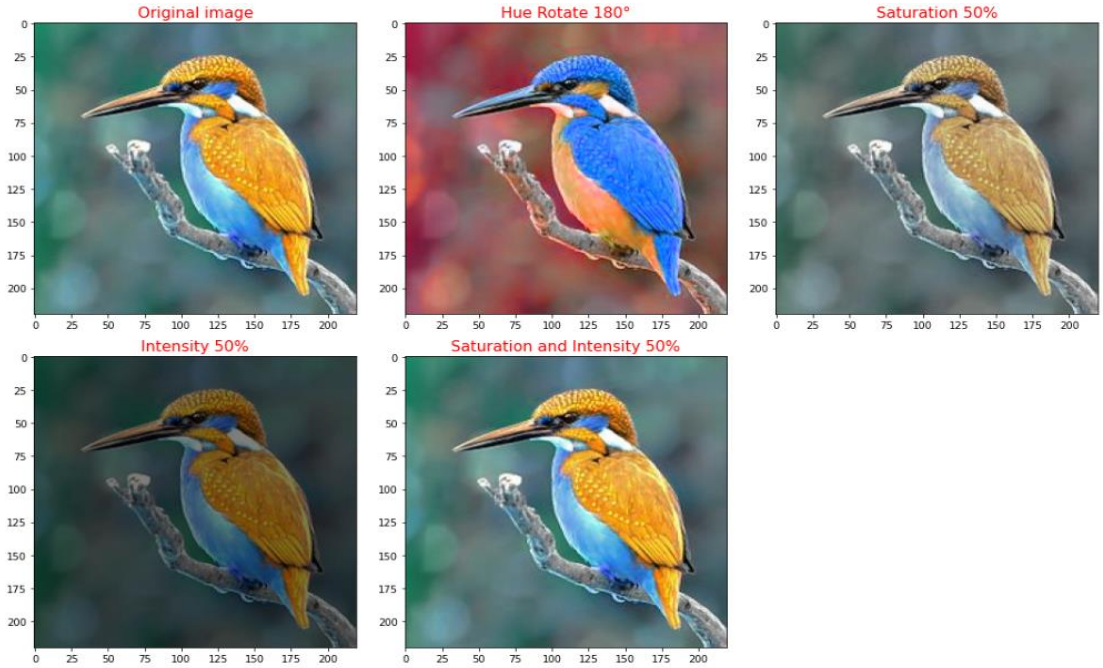

從上圖可以看出 HSV 直方圖所輸出的影像比 RGB 直方圖來的要好，但都不及高斯濾波。

## 實戰四

```
import numpy as np
import cv2
import math
from matplotlib import pyplot as plt
from google.colab import drive
from google.colab.patches import cv2_imshow

drive.mount('/content/drive')
img = cv2.imread("/content/drive/My Drive/Colab Notebooks/image_processing/week14.jpg", -1)

def RGB_to_HSI(R, G, B):
        r = R / 255
        g = G / 255
        b = B / 255

        if R == G and G == B:
                H = -1.0
                S = 0.0
                I = (r + g + b) / 3
        else:
                x = (0.5 * ((r - g) + (r - b))) / \
                        np.sqrt((r - g) ** 2 + (r - b) * (g - b))

                if x < -1.0:
                        x = -1.0
                if x > 1.0:
                        x = 1.0

                theta = np.arccos(x) * 180 / np.pi

                if B <= G:
                        H = theta
                else:
                        H = 360.0 - theta

                S = 1.0 - 3.0 / (r + g + b) * min(r, g, b)
                I = (r + g + b) / 3

        return H, S, I
```

```python
def HSI_to_RGB(H, S, I):
    if H == -1.0:
        r = I
        g = I
        b = I
    elif H >= 0 and H < 120:
        HH = H
        b = I * (1 - S)
        r = I * (1 + (S * np.cos(HH * np.pi / 180)) /
                 np.cos((60 - HH) * np.pi / 180))
        g = 3.0 * I - (r + b)
    elif H >= 120 and H < 240:
        HH = H - 120.0
        r = I * (1 - S)
        g = I * (1 + (S * np.cos(HH * np.pi / 180)) /
                 np.cos((60 - HH) * np.pi / 180))
        b = 3 * I - (r + g)
    else:
        HH = H - 240
        g = I * (1 - S)
        b = I * (1 + (S * np.cos(HH * np.pi / 180)) /
                 np.cos( (60 - HH) * np.pi / 180))
        r = 3 * I - (g + b)

    rr = round(r * 255)
    gg = round(g * 255)
    bb = round(b * 255)
    R = np.uint8(np.clip(rr, 0, 255))
    G = np.uint8(np.clip(gg, 0, 255))
    B = np.uint8(np.clip(bb, 0, 255))

    return R, G, B
```

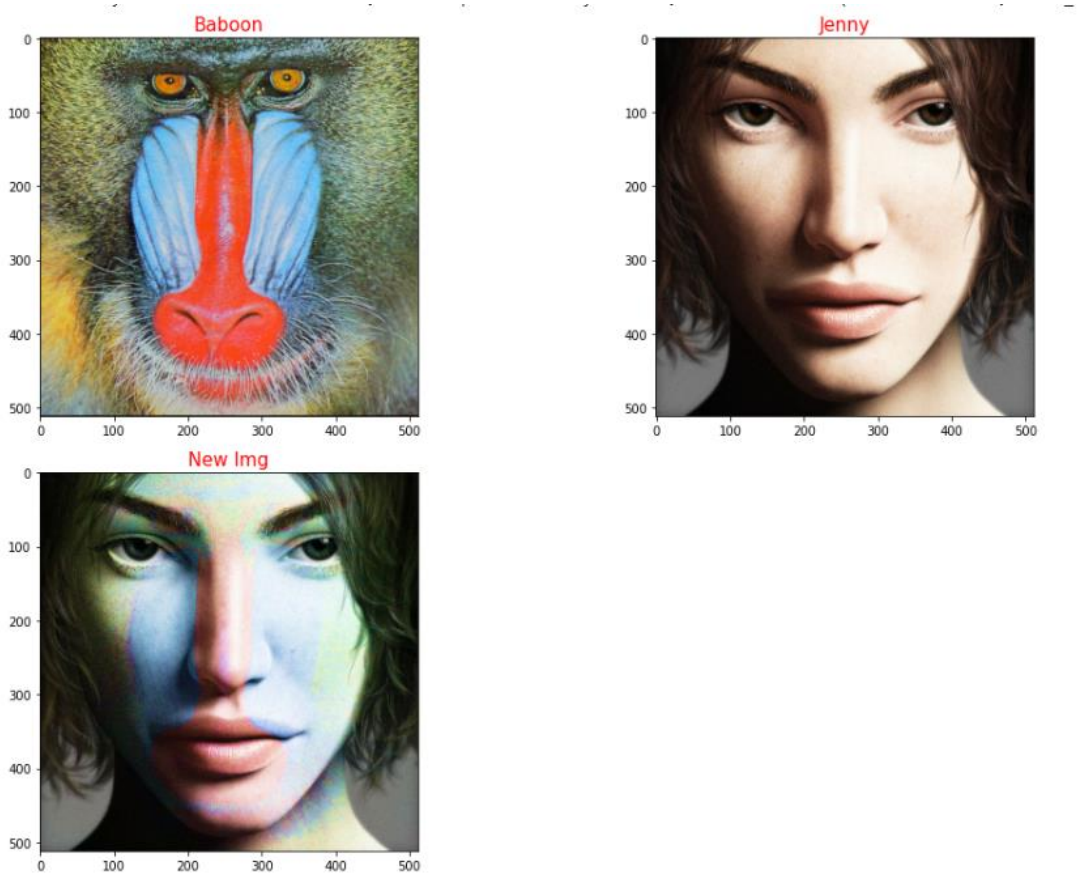
```python
def HSI_processing(f, angle = 0, saturation = 100, intensity = 100):
    g = f.copy()
    nr, nc = f.shape[:2]

    for x in range(nr):
        for y in range(nc):
            H, S, I = RGB_to_HSI(f[x, y, 2], f[x, y, 1], f[x, y, 0])
            H = H + angle

            if H > 360:
                H = H - 360

            S = S * saturation / 100
            I = I * intensity / 100
            R, G, B = HSI_to_RGB(H, S, I)

            g[x,y,0] = B
            g[x,y,1] = G
            g[x,y,2] = R

    return g

img1 = HSI_processing(img, 180, 100, 100)
img2 = HSI_processing(img, 0, 50, 100)
img3 = HSI_processing(img, 0, 100, 50)
img4 = HSI_processing(img, 0, 100, 100)

imgs = [img, img1, img2, img3, img4]
titles = ['Original image', 'Hue Rotate 180°', 'Saturation 50%', 'Intensity 50%', 'Saturation and Intensity 50%']

plt.figure(figsize = (15, 10))

for i in range(5):
    plt.subplot(2, 3, i + 1)
    plt.title(titles[i], fontsize = 15, color = 'r')
    plt.imshow(imgs[i], cmap = 'gray')

plt.tight_layout()
plt.show()
```




當色調旋轉 180°後，可以發現鳥的身體顏色對調了，而當飽和度調為 50%時，可以發現到整張影像的色彩偏灰黯，並不是特別的鮮豔，而當強度調為 50%時，可以發現到整張影像亮度變暗了，而當飽和度和強度調為 100%時，則會跟原圖所呈現的想同。

# 實戰五

```python
import numpy as np
import cv2
import math
from matplotlib import pyplot as plt
from google.colab import drive
from google.colab.patches import cv2_imshow

drive.mount('/content/drive')
Baboon = cv2.imread("/content/drive/My Drive/Colab Notebooks/image_processing/Baboon.bmp", -1)
Jenny = cv2.imread("/content/drive/My Drive/Colab Notebooks/image_processing/Jenny.bmp", -1)

def combin(h, s, v):
    nr, nc = h.shape
    hsv = np.zeros([nr, nc, 3], dtype = 'uint8')
    for x in range(nr):
        for y in range(nc):
            hsv[x, y, 0] = h[x, y]
            hsv[x, y, 1] = s[x, y]
            hsv[x, y, 2] = v[x, y]

    return hsv

Baboon_HSV = cv2.cvtColor(Baboon, cv2.COLOR_BGR2HSV)
B_H, B_S, B_V = cv2.split(Baboon_HSV)
Baboon = cv2.cvtColor(Baboon, cv2.COLOR_BGR2RGB)

Jenny_HSV = cv2.cvtColor(Jenny, cv2.COLOR_BGR2HSV)
J_H, J_S, J_V = cv2.split(Jenny_HSV)
Jenny = cv2.cvtColor(Jenny, cv2.COLOR_BGR2RGB)

NewImg = combin(B_H, J_S, J_V)
NewImg = cv2.cvtColor(NewImg, cv2.COLOR_HSV2RGB)

imgs = [Baboon, Jenny, NewImg]
titles=['Baboon', 'Jenny', 'New Img']

plt.figure(figsize = (15, 10))

for i in range(3):
        plt.subplot(2, 2, i + 1)
        plt.title(titles[i], fontsize = 15, color = 'r')
        plt.imshow(imgs[i], cmap = 'gray')

plt.tight_layout()
plt.show()
```




通過上圖可知，Jenny 將色調(H)換成 Baboon 的後，她的膚色就變得跟 Baboon 的膚色一樣了

## 問答題

(a) 已知哆啦 A 夢身體的顏色是青色 (Cyan)，鈴鐺是黃色 (Yellow)，

若使用紅色光照射時，請問哆啦 A 夢的身體與鈴鐺，會呈現何種顏色，並說明其原因

青色光是由藍光和綠光所組成的，因此只會反射藍光和綠光，也因此使用紅光照射時，紅光被吸收，沒有任何光線反射，因而呈現黑色。
而黃色光是由紅光和綠光所組成的，因此只會反射紅光和綠光，也因此使用紅光照射時，綠光被吸收，反射紅光，因而呈現紅色。
由上述可知，在紅光的照射下哆啦 A 夢的身體會呈現黑色，鈴鐺會呈現紅色

(b) 已知 HSI 的數值如下，試判斷是屬於何種顏色?

- H, S, I = 60, 0.8, 0.7

  A: 黃色

- H, S, I = 120, 0.8, 0.8

  A: 綠色

- H, S, I = 240, 0.8, 0.6

  A: 藍色

(c) 試說明色彩影像的影像濾波，應如何進行?

將 RGB 三通道視為獨立的數位影像，再分別套用影像濾波技術，例如使用平均濾波、高斯濾波