HW 02 - Histogram equalization

```python
# Let's try to work on the 3 channels.
# Using a weighted grayscale, getting y as intensity, and then apply equalization.

# split the color image into 3 channels
# Note: after splitting, the output will be in the reverse order of the input,
# i.e., input RGB -> output BRG

(b, g, r) = cv.split(image)
y = 0.3*r + 0.59*g + 0.11*b  # Weighted grayscale, getting y as intensity
y = y.astype(np.uint8)  # change float to 0-255 integer values
cv2_imshow(y)
print("y (intensity); y = 0.3*r + 0.59*g + 0.11*b \n")

y_equalized = cv.equalizeHist(y)
cv2_imshow(y_equalized)
print("y (intensity) after HE\n")
```

y (intensity); y = 0.3*r + 0.59*g + 0.11*b

y (intensity) after HE

จากโค้ด จะเป็นการนำค่า RGB (สี 3 channel) ของรูปภาพมาใส่ตัวแปร b, g, r อยู่ในรูป metric แล้วปรับค่า b, g, r ให้เป็นภาพ Gray Scale (สี 1 channel) ด้วยการให้น้ำหนักค่าแต่ละสีต่างกันตามสมการ Y = 0.3*R + 0.59*G + 0.11*B เมื่อได้ค่า Y เป็นทศนิยม (float) นำมาแปลงค่าให้เป็นจำนวนเต็มในช่วงค่า 0 – 255 แล้วภาพที่ปรับเป็น Gray Scale มาเฉลี่ยค่า histogram ให้กระจายเท่าๆ กัน หรือการทำ Histogram Equalization (cv.equalizeHist) ในแต่ละช่วงสีตั้งแต่ 0 – 255 เพื่อให้ภาพไม่มีค่า histogram ตกอยู่ในช่วงสีช่วงใดช่วงหนึ่งมากเกินไป พบว่า ภาพผลลัพธ์จะมองเห็นรายละเอียดได้ง่ายกว่าภาพต้นฉบับ

เปรียบเทียบกับโค้ด Histogram Equalization ในส่วนอื่น

```python

# Now, do a histogram equalization

equ = cv.equalizeHist(image_gray)   # input is a one-channel grayscale image

# Create a blank white image, and use it as a viewing separator
img_white = np.zeros([image_gray.shape[0],image_gray.shape[1]],dtype=np.uint8) # used for 1 channel
img_white[:] = 255

res = np.hstack((image_gray, img_white, equ)) #stacking images side-by-side, for a clear comparison
cv2_imshow(res)
print("\t\tGray \t\t vs. \t\t\t Histogram Equalized")
```

Gray vs. Histogram Equalized

จากโค้ดในส่วนนี้จะเป็นการทำภาพ Gray Scale ด้วยการใช้โค้ด cv.COLOR_BGR2GRAY แล้วนำมาทำ Histogram Equalization ด้วยวิธีการเดียวกันกับโค้ดชุดแรก (cv.equalizeHist) ซึ่งผลลัพธ์ของภาพหลังปรับ histogram แล้ว มองด้วยตาเปล่าแล้วไม่แตกต่างกัน

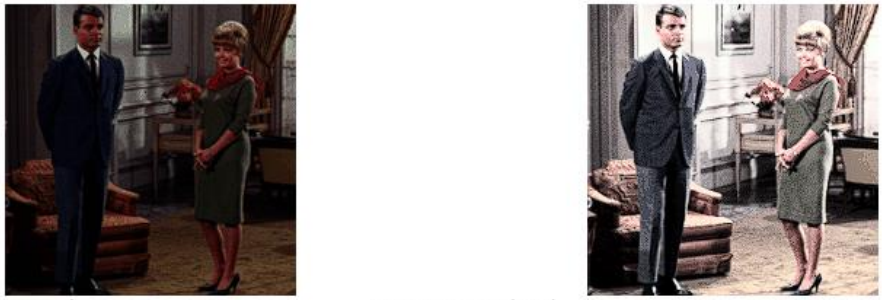
ในโค้ดส่วนอื่นๆ จะเป็นการทำ Histogram Equalization ด้วยวิธีการเดียวกันกับโค้ดชุดแรกเช่นเดียวกัน (cv.equalizeHist) แต่แตกต่างกันที่ input และ output เป็นภาพสี (สีมากกว่า 1 channel)

```python
# convert it to YCrCb color space
img_ycrcb = cv.cvtColor(image,cv.COLOR_RGB2YCrCb)

# apply histogram equalization
img_ycrcb[:,:,0] = cv.equalizeHist(img_ycrcb[:,:,0])  # use only Y channel
hist_eq = cv.cvtColor(img_ycrcb, cv.COLOR_YCrCb2RGB)

# Create a blank white image, and use it as a viewing separator
img_white = np.zeros([image.shape[0],image.shape[1],3],dtype=np.uint8)
img_white[:] = 255

# display both images (original and equalized)
cv2_imshow(np.hstack((image,img_white,hist_eq)))   # the hstack() is used to stack both images
print("Original \t\t\tvs.\t\t\t Histogram Equalizied\n")
```



Original vs. Histogram Equalizied

```python
# Histogram Equalization for color image

# convert it to YUV color space
img_yuv = cv.cvtColor(image,cv.COLOR_RGB2YUV)

# apply histogram equalization
img_yuv[:,:,0] = cv.equalizeHist(img_yuv[:,:,0])  # use only Y channel
hist_eq = cv.cvtColor(img_yuv, cv.COLOR_YUV2RGB)

# display both images (original and equalized)
cv2_imshow(np.hstack((image, img_white, hist_eq)))   # the hstack() is used to stack both images
```

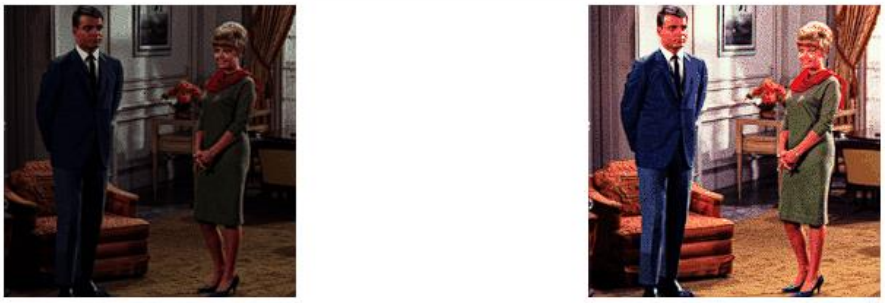
สำหรับโค้ด 2 ชุดนี้เป็นการทำ Histogram Equalization ภาพสีเหมือนกัน แต่จะทำการแปลงสีเป็นมาตรฐานสีที่แตกต่างกัน โค้ดชุดแรกจะแปลงจากจาก RGB เป็น YCbCr โค้ดชุดที่สองจะแปลงจาก RGB เป็น YUV แล้วกระจายเพียง Y channel ในการทำ Histogram Equalization เหมือนกันทั้งสองชุด แล้วแปลงสีกลับเป็น RGB ผลลัพธ์ของภาพจากโค้ดทั้งสองชุดมองด้วยตาเปล่าแล้วไม่แตกต่างกัน

```
[ ]
# convert it to HSV color space
# It's not recommended to the use of V as it is a brightness but not an intensity

img_hsv = cv.cvtColor(image,cv.COLOR_RGB2HSV)

# apply histogram equalization
img_hsv[:,:,2] = cv.equalizeHist(img_hsv[:,:,2])  # use only V channel
hist_eq = cv.cvtColor(img_hsv, cv.COLOR_HSV2RGB)

# display both images (original and equalized)
cv2_imshow(np.hstack((image, img_white, hist_eq)))   # the hstack() is used to stack both images
```



โค้ดชุดนี้คล้ายคลึงกับโค้ดสองชุดก่อนหน้า นั่นคือ เป็นการทำ Histogram Equalization ภาพสี แต่โค้ดชุดนี้จะทำการแปลงสีจาก RGB เป็น HSV แทน แล้วนำเข้า cv.qualizeHist ด้วยการกระจายเพียง V channel แล้วแปลงจาก HSV กลับเป็น RGB พบว่า ภาพผลลัพธ์จะมีสีที่เข้มกว่าภาพจากโค้ดสองชุดก่อนหน้า

```python
# Interesting note on HE and RGB
# Not a proper way to do, due to using R,G,B as mentioned earlier
# Therefore, we should not perform the HE on each color, separately.

def histogram_equalize(img):
    b, g, r = cv.split(img)
    red = cv.equalizeHist(r)
    green = cv.equalizeHist(g)
    blue = cv.equalizeHist(b)
    return cv.merge((blue, green, red))

incorrect_img = histogram_equalize(image)
cv2_imshow(image)
print("Original")
cv2_imshow(incorrect_img)
print("Incorrect, due to the misuse of RGB")
```

Original

Incorrect, due to the misuse of RGB

ในส่วนของโค้ดชุดสุดท้าย เป็นการทำ Histogram Equalization ในแต่ละ channel สีของ RGB เป็นการกระจาย histogram ของแต่ละสี ทำให้สีของภาพผลลัพธ์ดูสว่างขึ้น แต่เป็นกระบวนการที่ไม่ควรทำ เนื่องจากการปรับ histogram แต่ละ channel แยกกัน จะทำให้ภาพรวมการปรับ histogram ออกมาไม่ถูกต้อง