Lab2-2 : Image Enhancement with Statistical Operation

Histogram Equalization

[34]:

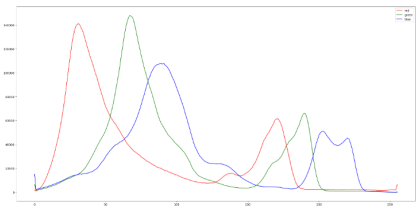
```
### START CODE HERE ###
def plot_histogram(image):
    colors = ['red', 'green', 'blue']
    hists = []
    for i,color in enumerate(colors):
        # print(f"i: {i}, color: {color}")
        hists.append(cv2.calcHist([image],[i],None,[256],[0,256]))
    return hists
### END CODE HERE ###
```

- Return ออกมาเป็น List ของ Histogram ที่มีขนาดเป็น 3 โดยจะเป็น Histogram ของสีแดง สีเขียว สีฟ้า (RGB) ตามลำดับ
- ใช้ฟังชั่น cv2.calcHist() โดยจะรับ parameter เป็น image และ index ของสีที่สนใจ
- นอกจากนี้ยังรับ mask ที่เราไม่ได้ต้องการใช้ และก็เป็น size และ range ของ dimension สีของรูปภาพ

Before:

[36]:

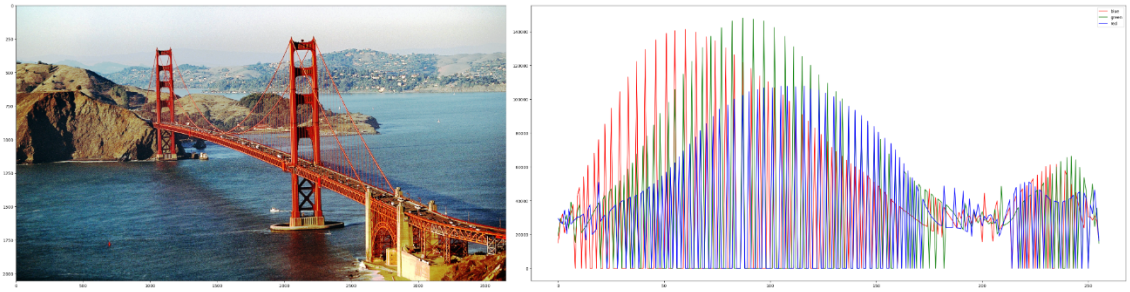
```
### START CODE HERE ###

figg = plt.figure(figsize=(40, 10))
axs = figg.subplots(1, 2)
# fig.figsize = [20, 20]
# print(fig.available())
figg.tight_layout()
axs[0].imshow(img)
hists = plot_histogram(img)
colors = ['red', 'green', 'blue']
i = 0
for hist in hists:
    plt.plot(hist, color=colors[i])
    i += 1
plt.legend(colors)
plt.show()


### END CODE HERE ###
```



- นำ list ของ Histogram data จากฟังชั่นข้างต้นมาแสดงทีละอัน โดยใช้สีที่แตกต่างกัน โดยใช้ฟังชั่น plt.ploy(hist, color=colors[i]) โดย hist คือ iterator ของ list และ colors = [‘red’, ‘green’, ‘blue’]

After:

[38]:
```
### START CODE HERE ###

eq_img = []
eq_img.append(cv2.equalizeHist(img[:, :, 0]))
eq_img.append(cv2.equalizeHist(img[:, :, 1]))
eq_img.append(cv2.equalizeHist(img[:, :, 2]))

combile_img = np.array([eq_img[0], eq_img[1], eq_img[2]])
combile_img = np.transpose(combile_img, (1, 2, 0))

figg = plt.figure(figsize=(40, 10))


axs = figg.subplots(1, 2)
axs[0].imshow(combile_img)

hists = plot_histogram(combile_img)
i = 0
for hist in hists:
    axs[1].plot(hist, color=colors[i])
    plt.tight_layout()
    i += 1

axs[1].legend(["red", "green", "blue"])
plt.show()


### END CODE HERE ###
```



- นำรูปภาพเดิม(ซึ่งเป็น array 3 มิติ) มาแยกเป็น array 2 มิติจำนวน 3 อัน เป็นของ channel R, G และ B ตามลำดับ จากนั้นนำแต่ละ Channel มาเข้าเป็นพารามิเตอร์ของฟังก์ชัน cv2.equalizeHist() เพื่อกระจายเฉดสีแต่ละสี

Image Histogram Matching

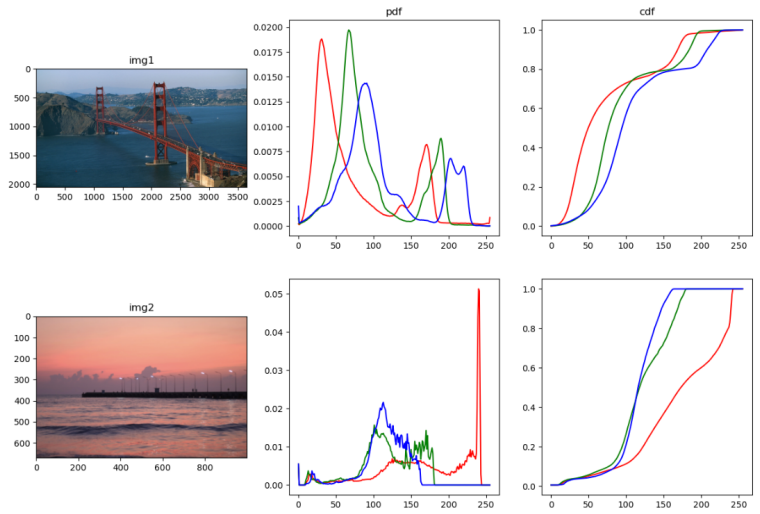
Load image:

```
[48]: ### START CODE HERE ###


img1 = cv2.imread("goldengate.jpg")
img2 = cv2.imread("image.png")


img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)

### END CODE HERE ###
```

- โหลดรูปภาพโดยใช้ฟังชั่น cv2.imread() แล้วนำมา convert จากที่ได้เป็น BGR ให้เป็น RGB ด้วยฟังก์ชัน cv2.cvtColor()

Display Histogram matching:

```
[69]: ### START CODE HERE ###

fcol = "white"
bcol = "black"
img_l = [img1, img2]
colors = ["red", "green", "blue"]

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

axs = figg.subplots(2, 3)

i = 0
axs[i][1].set_title("pdf")
axs[i][2].set_title("cdf")
for img in img_l:
    # axs[i][1].spines.color=fcol
    axs[i][0].set_title(f"img{i+1}")
    axs[i][0].imshow(img,)
    hists = plot_histogram(img)
    j = 0
    for hist in hists:
        hist = hist/hist.sum()
        axs[i][1].plot(hist, color=colors[j])
        cum_hist = hist.cumsum()
        # cum_hist = cum_hist/cum_hist.sum()
        axs[i][2].plot(cum_hist, color=colors[j])
        j += 1
    i += 1

### END CODE HERE ###
```

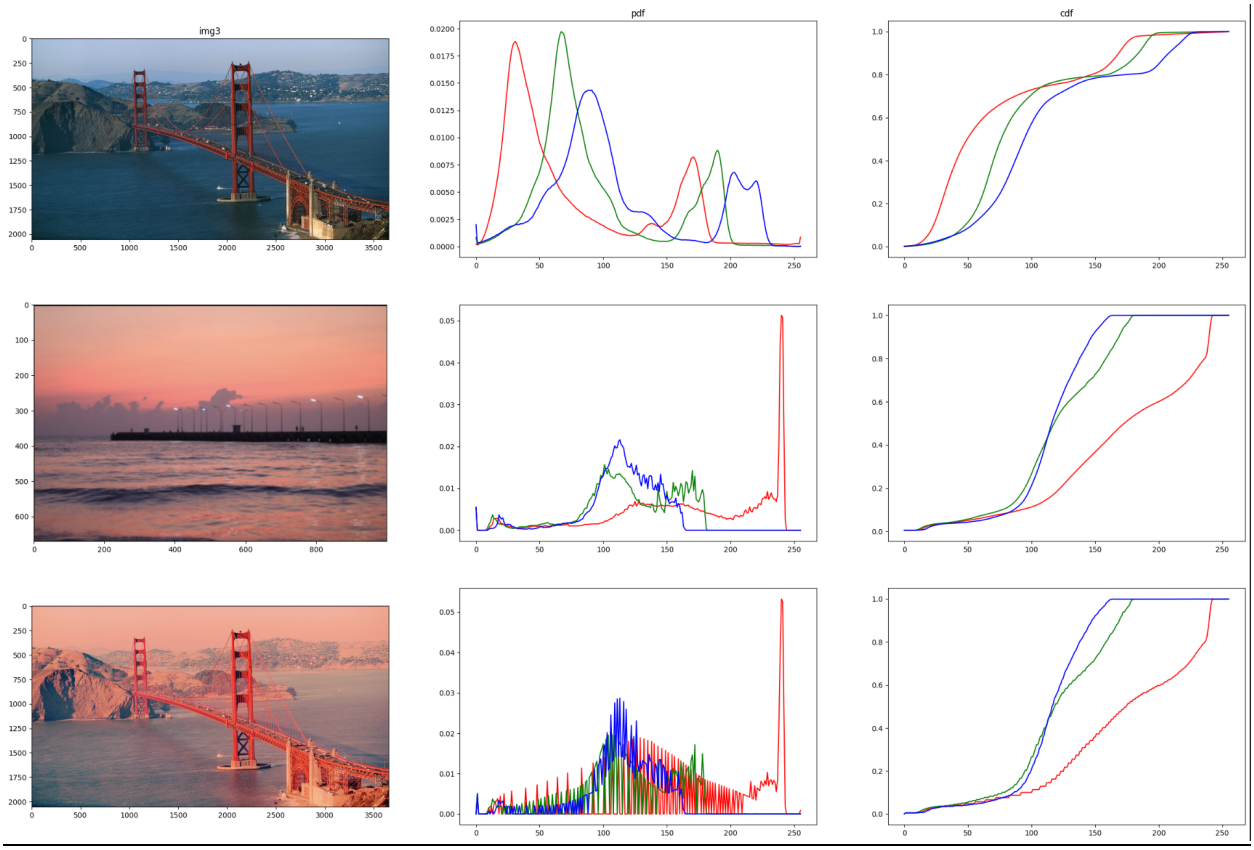


- ในการ display PDF เราสามารถ plot ได้โดยตรงจาก histrogram data ข้อก่อนหน้าได้เลย แต่ว่าในแนวแกน y เราต้องเป็นความน่าจะเป็น เราจึงต้องหารผลรวมของ histogram นั้นด้วย

- ส่วนในการ display CDF เราจำเป็นต้องนำมาเข้าฟังชั่น cumsum() เพื่อซึ่งฟังก์ชันนี้จะ return ออกมาเป็น list ใหม่ซึ่งคำนวณออกมาเป็น cumulative แล้ว ยกตัวอย่างเช่นหาก input เป็น [1,2,3,4] จะได้ output เป็น [1, 3, 6, 10]

Histrogram Matching

ผลลัพธ์:




```python
### START CODE HERE ###

def hist_matching(img1, cdf1, cdf2):
    map_hist = np.array([0] * len(cdf1))
    p_cdf2 = 0
    for p_cdf1 in range(len(cdf1)):
        while (p_cdf2 < len(cdf2) and cdf1[p_cdf1] > cdf2[p_cdf2]):
            p_cdf2 += 1
        if ((p_cdf2 >= len(cdf2) - 1) or (abs(cdf2[p_cdf2] - cdf1[p_cdf1]) < abs(cdf2[p_cdf2 + 1] - cdf1[p_cdf1]))):
            map_hist[p_cdf1] = p_cdf2
        else:
            map_hist[p_cdf1] = p_cdf2 + 1
    img3 = map_hist[img1]
    return (img3)
```

- โดยฟังก์ชั่นนี้จะเป็นฟังก์ชันหลักในการทำ histogram matching มีหลักการทำคือเราจำรับ cdf ทั้งของ input และ template มา และเราจะ loop cdf ของ input ไปเรื่อยๆ ว่า output ควรออกมาเป็นอะไร
- โดยทุกครั้งที่ loop เราจะเทียบกับตัว template ให้มันห่างกับ template น้อยสุด เช่น

    - หากเราลูปอยู่ที่ 0.23 แล้ว template เป็น [0.1, 0.2, 0.3, ...] เราก็จะเลือกตัวที่มีความห่างน้อยที่สุด ก็คือ 0.2
- Pseudo code
    - สร้างตัวแปรไว้เก็บค่า output ของ matching hist
    - Loop ทุก iterator ของ input cdf
        - ลูปเพื่อทำให้ iterator ของ template cdf เป็นค่ามากสุดที่ยังไม่เกิน iterator ของ input cdf
        - เช็คว่า iterator ของ template cdf กับ iterator + 1 (ตัวถัดไป) อันไหนมีความห่างกับ iterator ของ input cdf น้อยกว่ากัน แล้วเอาตัวนั้น
    - บรรทัดที่ 13 จะเป็นการนำ map มาลงที่ images โดย img1 จะทำหน้าที่เป็น index ซึ่งในกรณีนี้จะเป็นค่าสี และนำมาทำการ map กับ map_hist ซึ่งก็จะเป็นการอาแต่ละค่าสีของรูปจริง มาเทียบกับ map_hist แล้วเอาสีจาก map_hist นั้นไปแทน

```
for hist_1 in hists_1:
    pdf_1 = hist_1/hist_1.sum()
    axs[0][1].plot(pdf_1, color=colors[j])
    cdf_1.append(pdf_1.cumsum())
    axs[0][2].plot(cdf_1[j], color=colors[j])
    j+=1

j = 0
axs[1][0].imshow(img2)
cdf_2 = []
hists_2 = plot_histogram(img2)
for hist_2 in hists_2:
    pdf_2 = hist_2/hist_2.sum()
    axs[1][1].plot(pdf_2, color=colors[j])
    cdf_2.append(pdf_2.cumsum())
    axs[1][2].plot(cdf_2[j], color=colors[j])
    j += 1
```

- พล็อต 2 แถวแรก เหมือนข้อก่อนหน้า แต่ไม่ได้ทำเป็นลูปแล้ว

```
j = 0
img3 = np.array([hist_matching(img1[:,:, 0], cdf_1[0], cdf_2[0]), hist_matching(img1[:,:, 1], cdf_1[1], cdf_2[1]), hist_matching(img1[:,:, 2], cdf_1[2], cdf_2[2])])
img3 = np.transpose(img3, (1, 2, 0))
img3 = np.clip(img3, a_min = 0, a_max=255)
img3.astype(np.uint8)
axs[2][0].imshow(img3)
print(img3.shape)
img3 = cv2.convertScaleAbs(img3)

hists_3 = plot_histogram(img3)
for hist_3 in hists_3:
    pdf_3= hist_3/hist_3.sum()
    axs[2][1].plot(pdf_3, color=colors[j])
    cdf_3 = pdf_3.cumsum()
    axs[2][2].plot(cdf_3, color=colors[j])
    j += 1
```

- รับรูปภาพที่ผ่านการทำ histogram matching มาโดยเราจะ slice ทีละ Channel เนื่องจากฟังก์ชั่นที่เราใช้ matching เราทำให้มันสามารถรับได้แค่ 2 มิติ และรวมกันด้วย np.array() ซึ่งเป็น contructor ของ numpy array
- พอเรารับมาแล้วนำมารวมกัน มันจะได้ shape เป็น (COLOR, HEIGHT, WIDGTH) เราจึงต้องนำมา transpose ด้วย np.transpose และใช้พารามิเตอร์เป็น (1,2,0) เพื่อให้ได้ shape เป็น (HEIGHT, WIDGTH, COLOR)
- เนื่องจาก img3 มี depth ที่ไม่ตรงกับที่ plt.imshow() กับ calcHist() รับได้ เราจึงต้องปรับ img ก่อนโดยใช้ฟังก์ชัน np.clip() และ cv2.convertScaleAbs()