# پردازش تصویر( دکتر ساجدی )– تمرین اول

فرشاد برجعلی زاده – ۶۱۰۳۹۹۰۱۵

**در تمرین قصد داریم بر روی عکس های داده شده یک سری عملیات مقدماتی از جمله نحوه خواندن داده تصویر به صورت رنگی و سیاه و سفید، اضافه کردن بایاس به مقادیر پیکسل های تصویر، عملیات** sampling and quantization, scailing, translation, rotation, Histogram equalization **را انجام دهیم.**

**در این مجموعه داده ۳ عکس از گل و پروانه وجود دارد**



**در ابتدا کتابخانه های مورد نظر را وارد میکنیم**

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

**سپس با استفاده از تابع** imread() **از کتابخانه** cv2 **تصویر را میخوانیم و اطلاعات مربوط به آن را از جمله اندازه، بعد و نوع داده را نمایش میدهیم**

```
#Read original image
img = cv2.imread('image1.jpg')
```

```
print("Image Properties")
print("- Number of Pixels: " + str(img.size))
print("- Shape/Dimensions: " + str(img.shape))
print("- Image Data Type: " + str(img.dtype))
```

```
Image Properties
- Number of Pixels: 151200
- Shape/Dimensions: (168, 300, 3)
- Image Data Type: uint8
```

**اگر به صورت عادی تصویر را نمایش دهیم با چنین تصویری مواجه خواهیم شد.**

**و علت آن جابجایی کانال های رنگی تصویر است زیرا در کتابخانه cv2 رنگ ها به صورت BGR خوانده میشود در صورتی که به صورت معمول ما کانال های رنگی را به صورت RGB میخوانیم به همین خاطر می بایستی برای نمایش کانال های رنگی را مطابقت دهیم، به دو روش میتوان این کار را انجام داد.**

```
# We can split the our image into 3 three channels
blue, green, red = cv2.split(img)
# Next, we merge the channels in order to build a new image
img1 = cv2.merge([red, green, blue])
plt.imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x14122932f88>
```

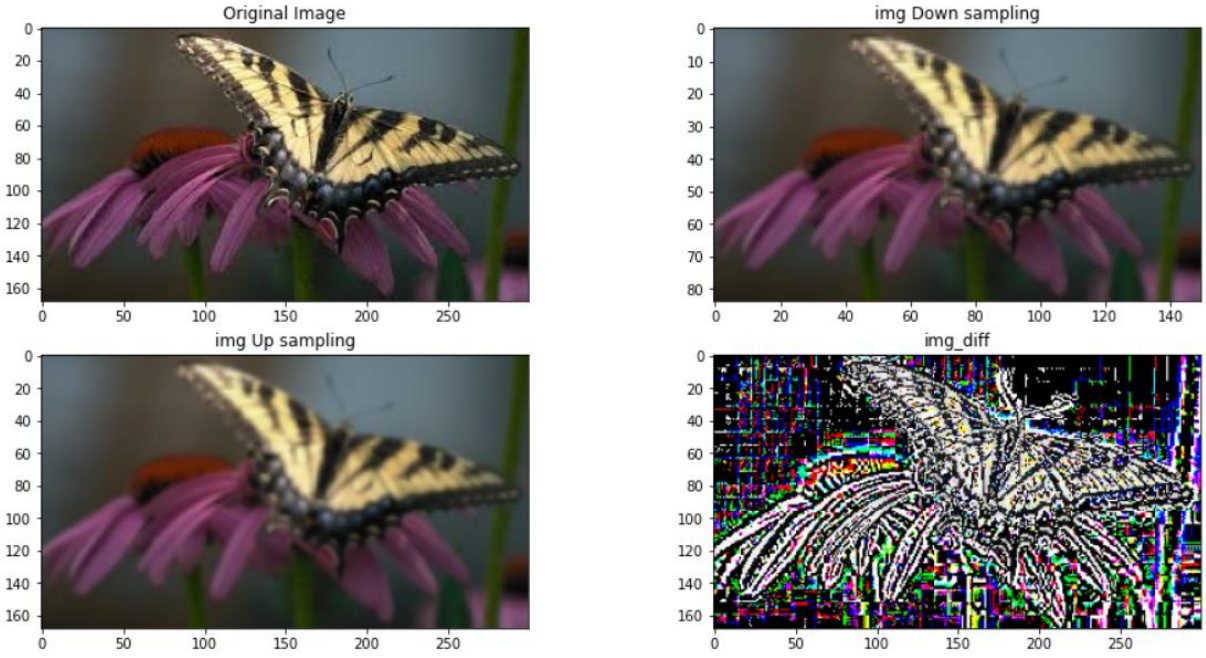
```
# Show original image with convert BRG channel to RGB
print('Original Image:')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

```
Original Image:

<matplotlib.image.AxesImage at 0x14123ee4448>
```

**عملیات** sampling **را به صورت انجام میدهیم یکی با استفاده از کتابخانه های موجود و دیگری به صورت دستی:**

**در روش با استفاده از کتابخانه های موجود** cv2.pyrDown **و** cv2.pyrUp **استفاده میکنیم که نتیجه آن بدین صورت است.**



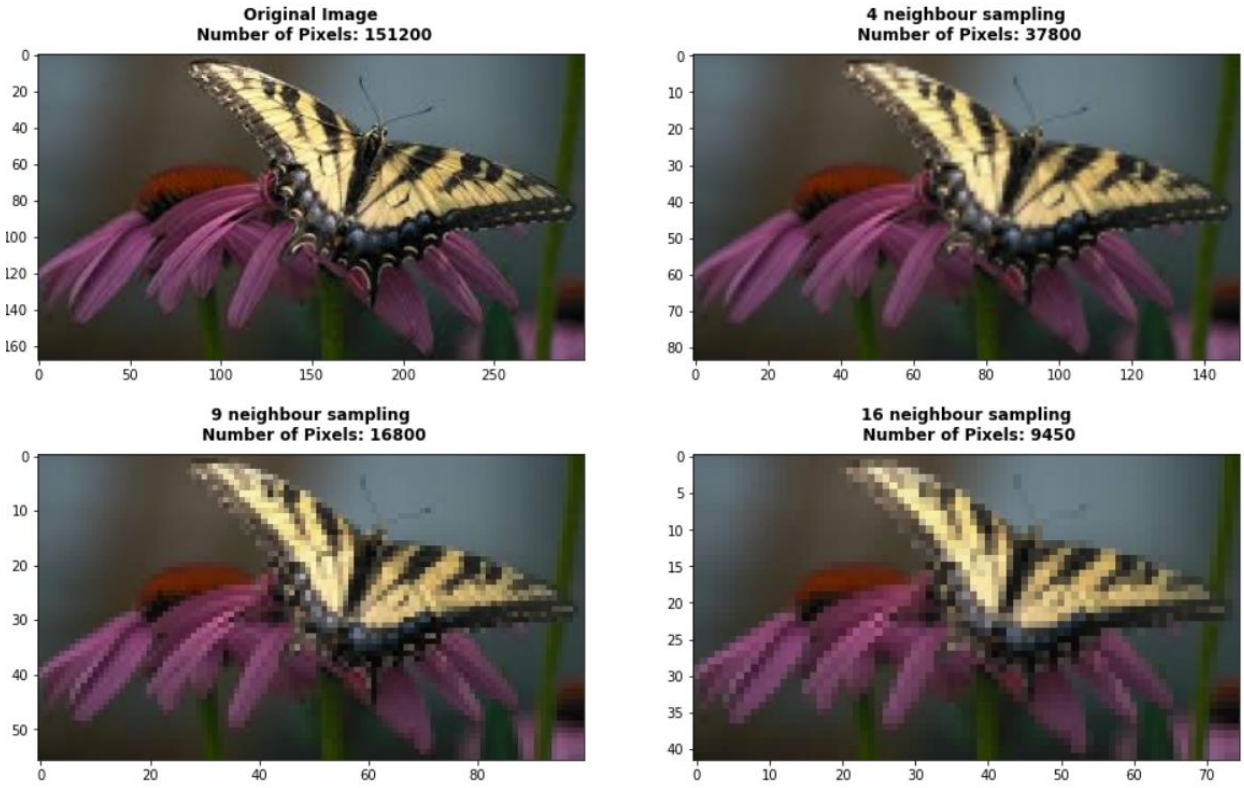

**تصویر سمت راست پایین از اختلاف دو تصویر** img Up sampling **و** img Down sampling **است**

**در روش دوم ابتدا یک ماتریس خالی تعریف میکنیم و سپس با استفاده از تعریف همسایگی های متفاوت آن را پر میکنیم.**

```
image2=np.zeros((int(image.shape[0]/ratio),
                 int(image.shape[1]/ratio),
                     image.shape[2]),dtype='float64')
```

```
for i in range(image2.shape[0]):
    for j in range(image2.shape[1]):
        for k in range(image2.shape[2]):
            delta=image[i*ratio:(i+1)*ratio,j*ratio:(j+1)*ratio,k]
            image2[i,j,k]=np.mean(delta)
```

**خروجی های تصویر اول را با همسایگی های متفاوت میبینیم.**



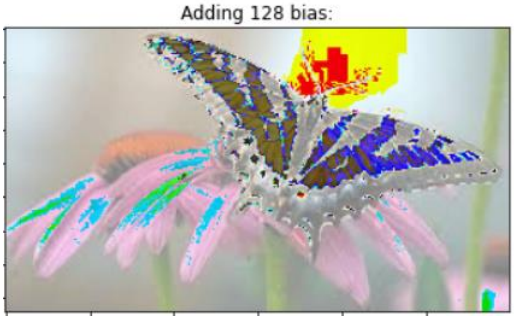

**همان طور که میبینیم با افزایش تعداد همسایگی ها تعداد پیکسل های عکس هم کاهش پیدا می کند.**

**در قسمت** quantization **با انتخاب** ratio=64 **و اعمال آن بر روی تصویر اصلی به تصویر زیر می رسیم.**

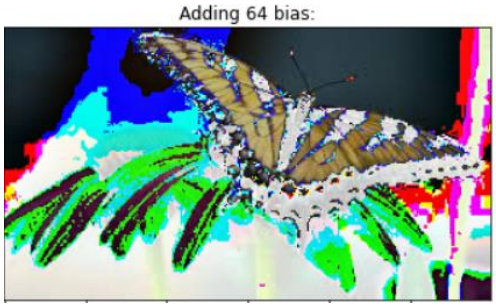
Image after quantification:

**برای اضافه کردن bias=128 ابتدا داده ها را به float64 تبدیل میکنیم و با اضافه کردن bias=128 و نرمال کردن دوباره داده ها تصویر مورد نظر بدین شکل می شود.**

Adding 128 bias:

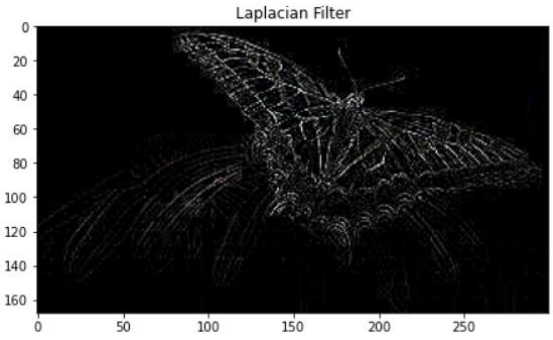


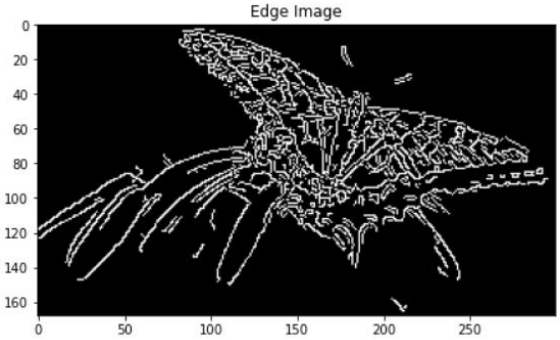
Adding 64 bias:



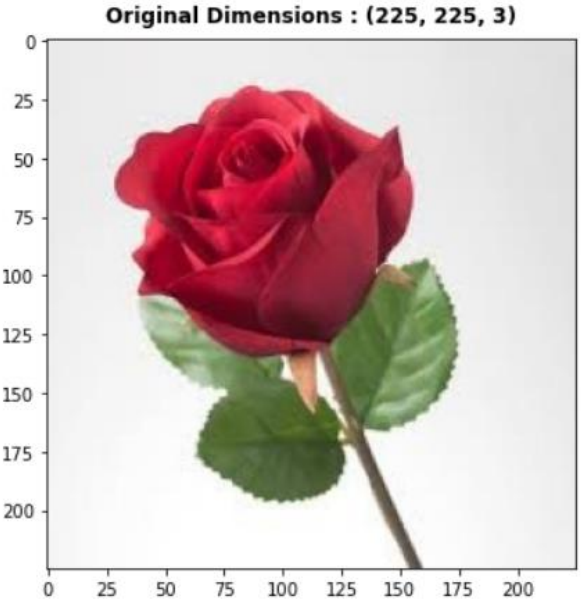
Adding 2 bias:

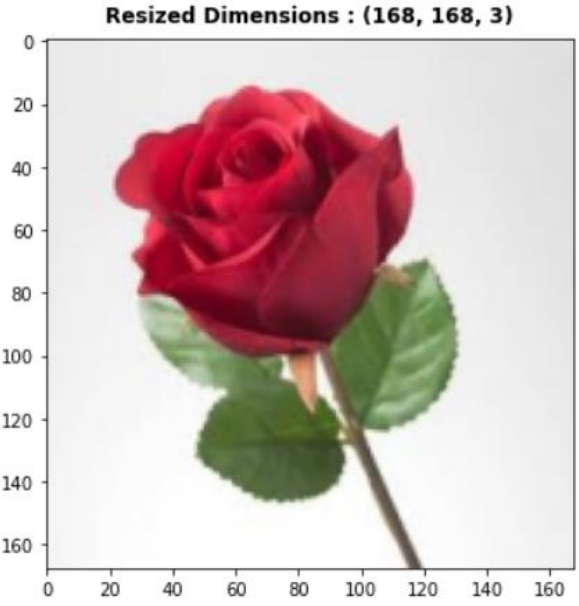
**بر روی تصویر اول علاوه بر مطالب خواسته شده دو فیلتر اضافی از جمله Laplacian و Canny هم اعمال شده که نتایج آن بدین صورت است.**



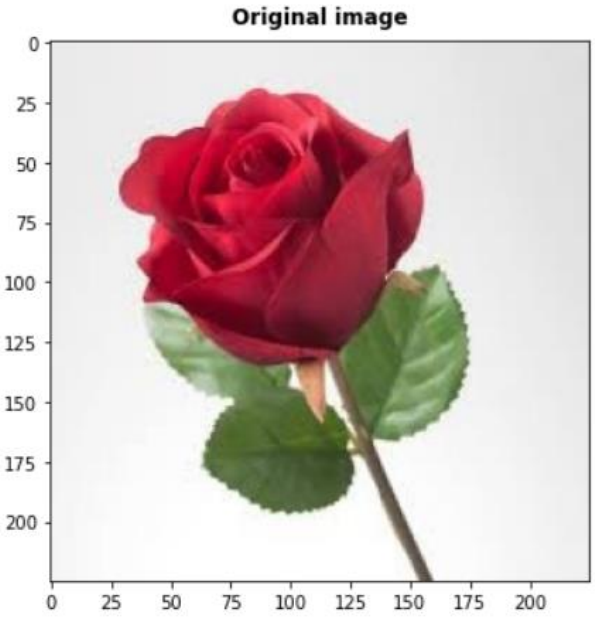

**در مورد قسمت مربوط به** Scailing **با خواندن تصویر دوم و اعمال** scale_percent = 75 **به تصویر مقابل دست پیدا کردیم.**

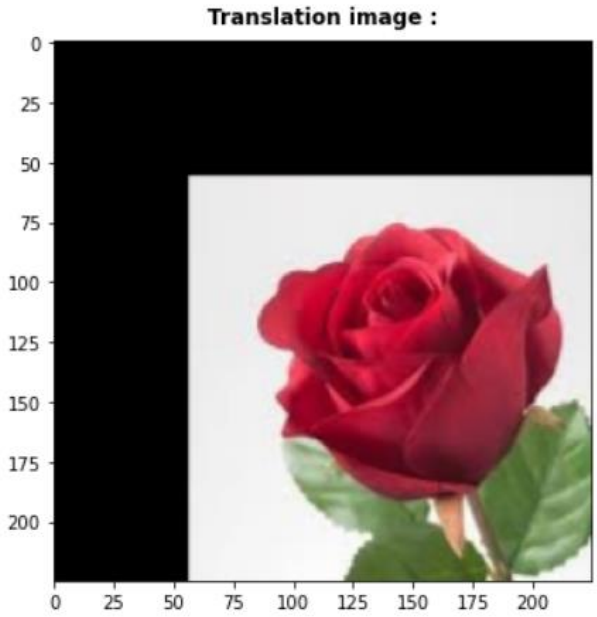


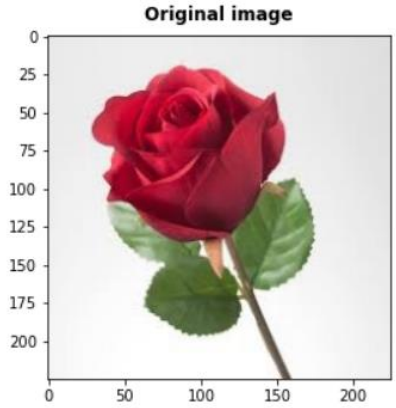



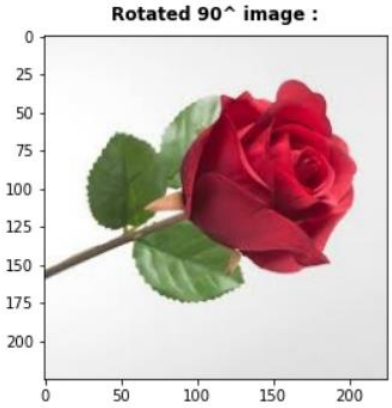

**که وضوح تصویر مقیاس شده به مراتب کمتر است.**

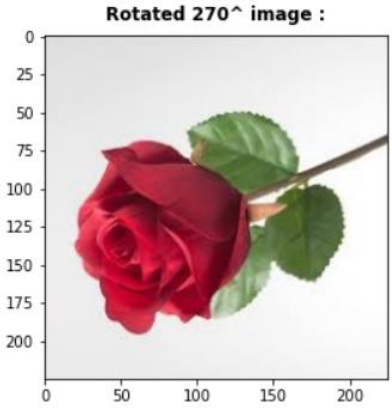
**نتایج مربوط به** Translation **و** Rotation **هم بدین صورت می باشد.**



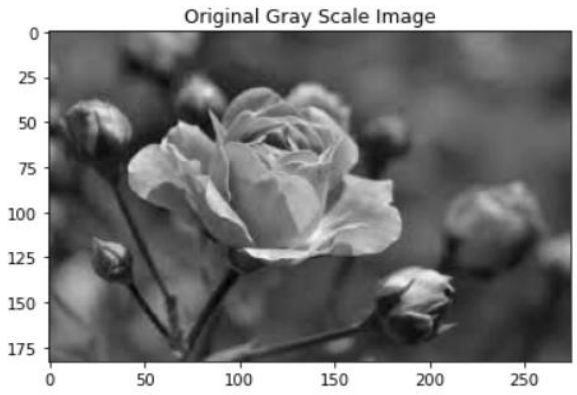



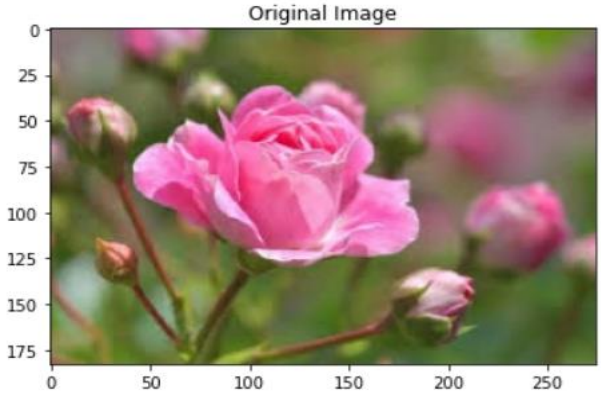



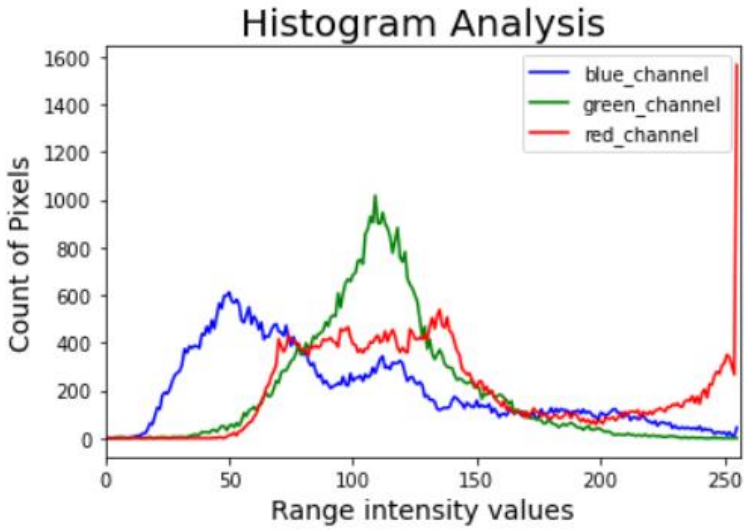



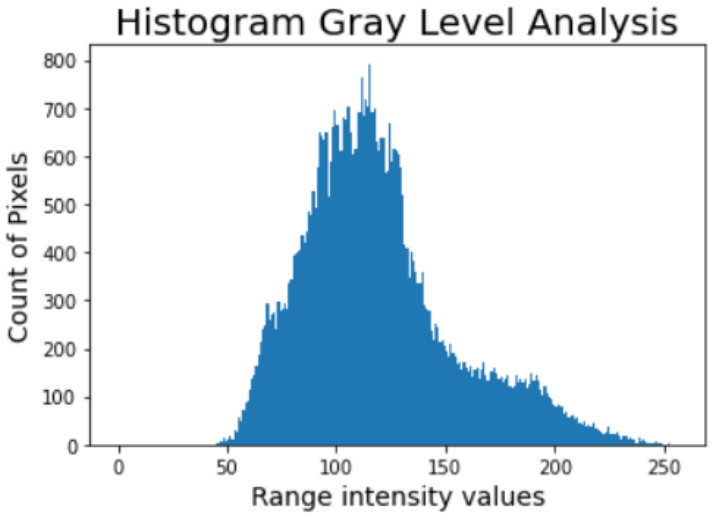

**در قسمت مربوط به هیستوگرام ابتدا عکس سوم را به صورت های رنگی و سیاه و سفید میخوانیم**



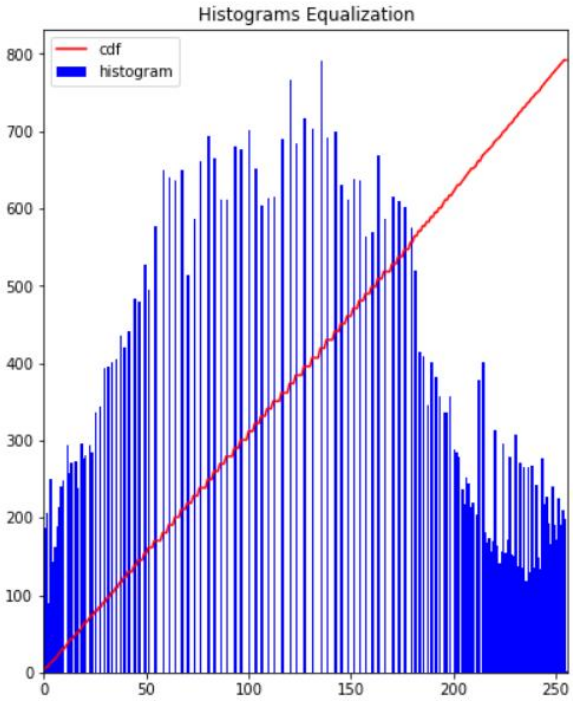



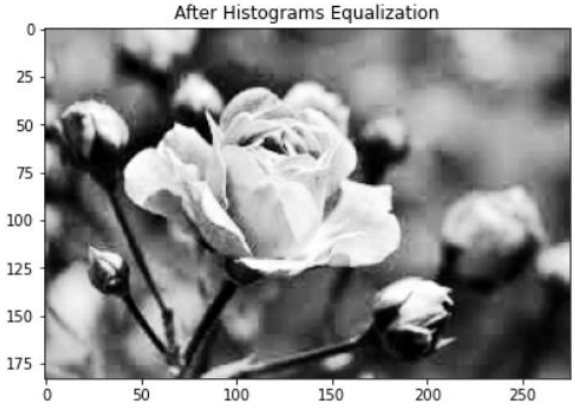
**سپس هیستوگرام های هر کدام از عکس ها را به تفکیک کانال های رنگی به نمایش در آورده ایم.**




**عملیات** histogram equalization **و** histogram matching transform **را فقط بر روی تصویر سیاه و سفید اعمال میکنیم که بعد از اعمال** histogram equalization **تابع و تصویر به این صورت تبدیل می شود.**




**سپس با اعمال** histogram matching transform **تصویر اول بر روی تصویر سوم به نتیجه مقابل دست پیدا می کنیم.**