5주차 - 화소 점 처리(3)

- 디지털 영상의 히스토그램(Histogram)
- 히스토그램 스트레칭(Histogram Stretching)
- 히스토그램 평활화(Histograms Equalization)
- 히스토그램 정합(Histogram Matching)

In [1]:
```
### Packages
import cv2
import numpy as np
from matplotlib import pyplot as plt
import os
```

In [2]:
```
### 출력 영상 크기
plt.rcParams["figure.figsize"] = (16,9)
### 한글 표시
plt.rcParams['font.family'] = "Gulim"   # 'AppleGothic' in mac
```

▶ 디지털 영상의 히스토그램(Histogram)

- 히스토그램: 데이터의 특징을 한 눈에 알아볼 수 있도록 데이터를 막대그래프로 나타낸 것
- 디지털 영상의 히스토그램: 각 화소들의 밝기 값의 발생 빈도로 이루어진 그래프
- 영상의 명도와 명암 대비를 파악
  - 왼쪽으로 치우침: 영상 밝기가 어두움
  - 오른쪽으로 치우침: 영상 밝기가 밝음
  - 좁은 범위에 분포: 명도 차이가 적어, 명암 대비가 좋지 않음
  - 넓은 범위에 분포: 명도 차이가 커, 명암 대비가 좋음

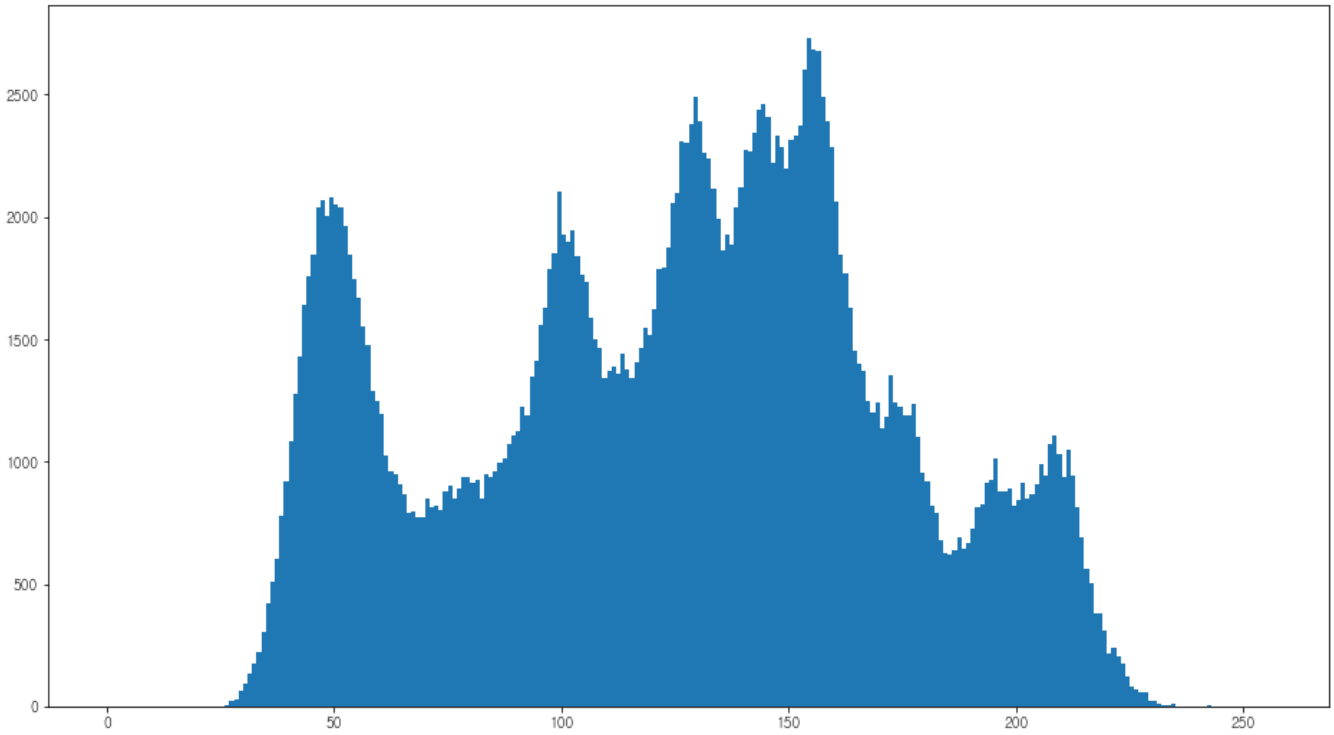
In [3]:
```
### 영상 읽기
img_lena = cv2.imread(r'D:WimageWlena.png')
img_lena.shape
```

Out[3]:
```
(512, 512, 3)
```

In [4]:
```
### Grayscale
img_lena_gray = cv2.cvtColor(img_lena, cv2.COLOR_BGR2GRAY)
img_lena_gray.shape
```

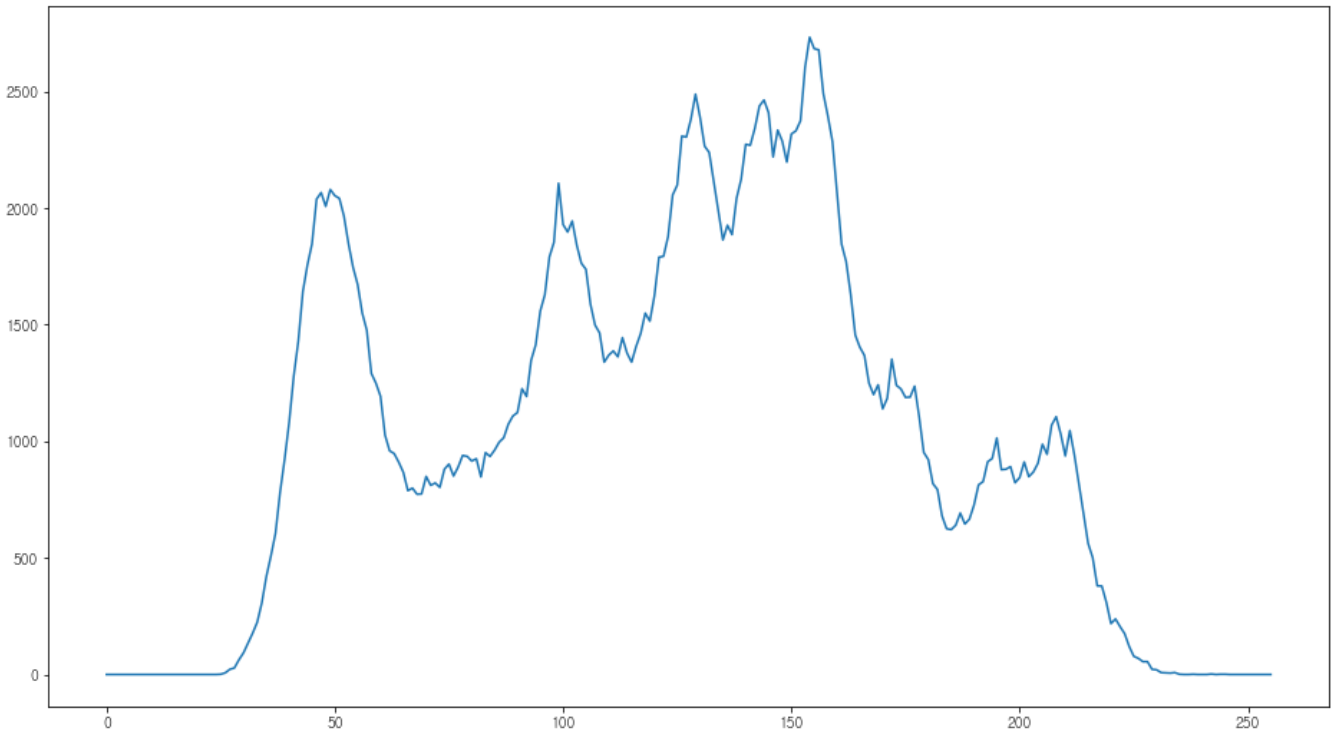
Out[4]:
```
(512, 512)
```

In [5]:
```
### 히스토그램 - grayscale
plt.hist(img_lena_gray.flatten(), 256, [0,256])
plt.show()
```



- cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])
  - image – 영상(uint8 or float32 type)
  - channels – 대상 채널graysacle이면 [0], color 이미지이면 [0],[0,1] 형태(1 : Blue, 2: Green, 3: Red)
  - mask – 대상 영역. None = 전체
  - histSize – BINS 값. [256]
  - ranges – 범위. [0,256]

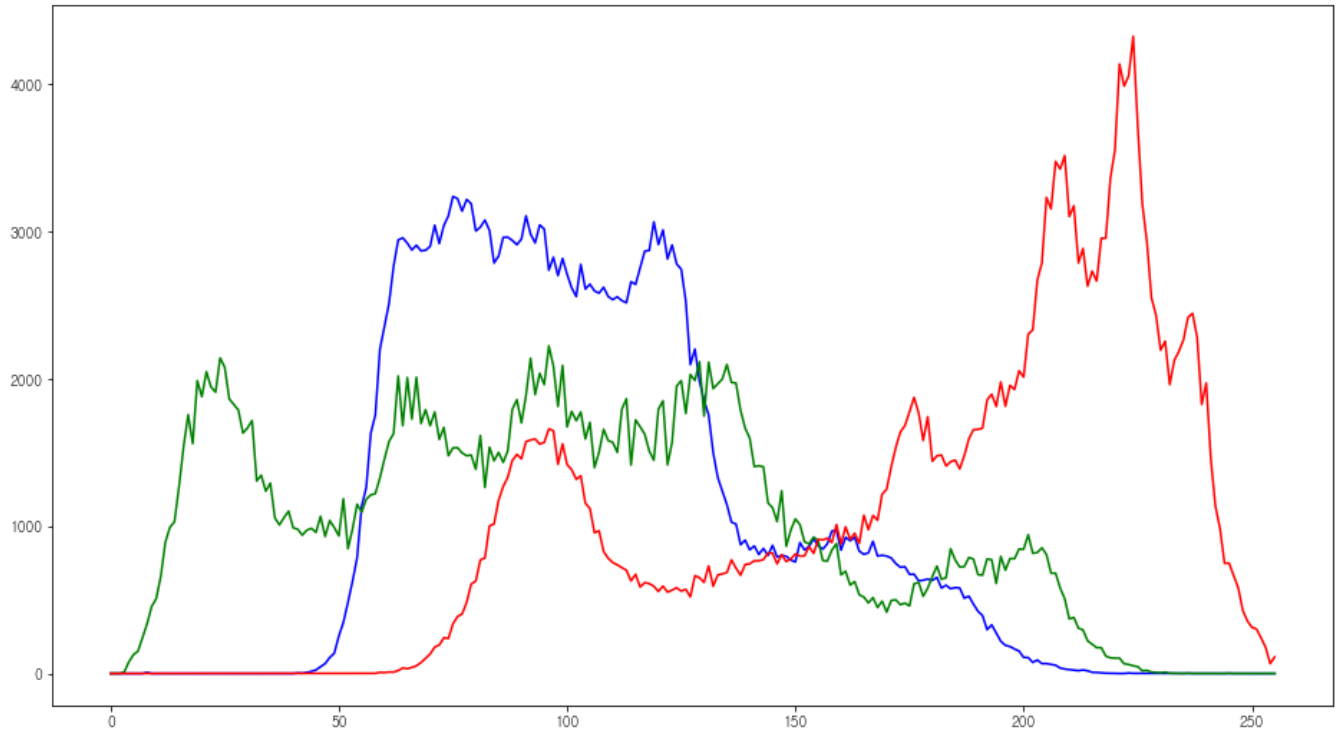
In [6]:
```
### 히스토그램 - grayscale
img_hist = cv2.calcHist([img_lena_gray], [0], None, [256], [0, 256])
plt.plot(img_hist)
plt.show()
```



In [7]:
```
### 히스토그램 - color
bgr = cv2.split(img_lena)
colors = ['b', 'g', 'r']

for ch, col in zip(bgr, colors):
```

```
    img_hist = cv2.calcHist([ch], [0], None, [256], [0, 256])
    plt.plot(img_hist, color=col)
plt.show()
```



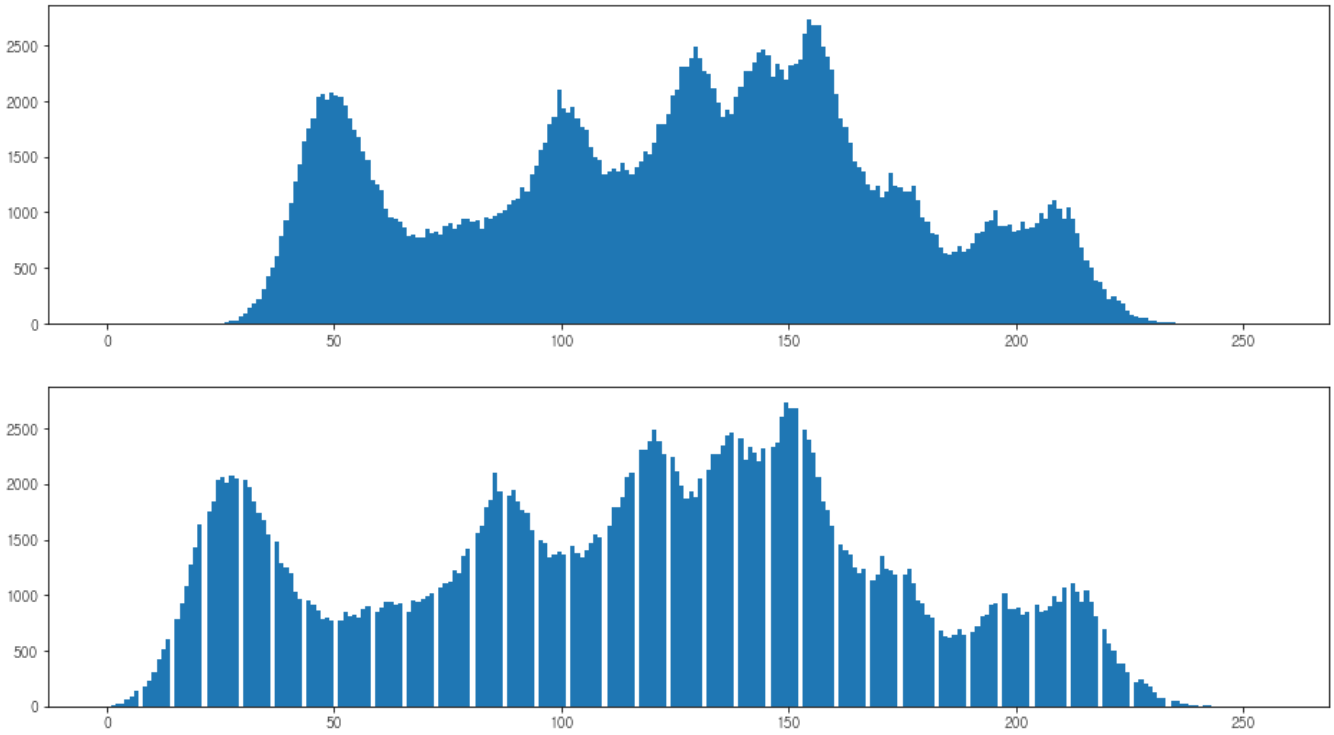
▶ 히스토그램 스트레칭(Histogram Stretching)

- 명암 대비를 향상시키는 연산으로 낮은 명암 대비를 보이는 영상의 품질을 향상시키는 방법
  - 히스토그램이 모든 범위의 화소 값을 포함하도록 분포를 넓힘

$$out = \frac{in - min}{max - min} \times 255$$

In [8]:
```
### 히스토그램 스트레칭 - grayscale
pixel_min = np.min(img_lena_gray)
pixel_max = np.max(img_lena_gray)
img_hist_stretching = np.array((img_lena_gray - pixel_min) /
                               (pixel_max - pixel_min) * 255).astype('uint8')
```

In [9]:
```
### 히스토그램 출력
plt.subplot(2, 1 , 1), plt.hist(img_lena_gray.flatten(), 256, [0,256])
plt.subplot(2, 1 , 2), plt.hist(img_hist_stretching.flatten(), 256, [0,256])
plt.show()
```



In [10]:
```
### 영상 출력
titles = ["원래 영상", "히스토그램 스트레칭"]
images = [img_lena_gray, img_hist_stretching]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(1, 2, i+1), plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```

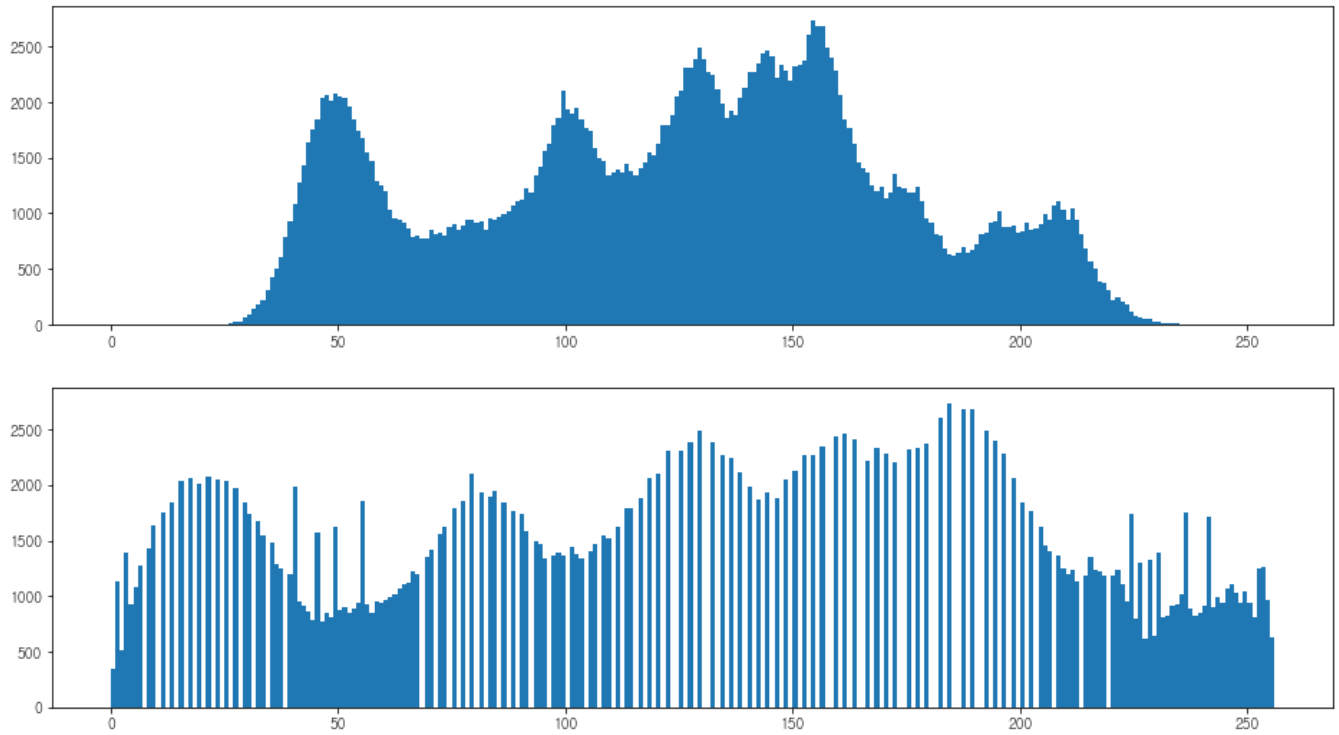
▶ 히스토그램 평활화(Histograms Equalization)

- 개념
  - 균일하지 않은 분포의 히스토그램을 균일하게 만드는 것
  - 히스토그램의 비선형적인 누적 분포 함수를 선형적인 형태로 변형
- 과정
  - 영상의 히스토그램 생성
  - 히스토그램 정규화

    - 정규화된 히스토그램의 누적분포함수를 이용한 대응 화소값 생성
    - 영상의 각 화소 값들로부터 대응 화소값 으로의 매핑
- 특징
    - 밝거나 어두운 영상을 사용해도 결과가 동일함
    - 이미지 인식을 할 때 유용함
- OpenCV - Histogram Equalization
    - https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html

In [11]:
```
### Histograms Equalization in OpenCV
img_hist_equal = cv2.equalizeHist(img_lena_gray)
```

In [12]:
```
### 히스토그램 출력
plt.subplot(2, 1 , 1), plt.hist(img_lena_gray.flatten(), 256, [0,256])
plt.subplot(2, 1 , 2), plt.hist(img_hist_equal.flatten(), 256, [0,256])
plt.show()
```



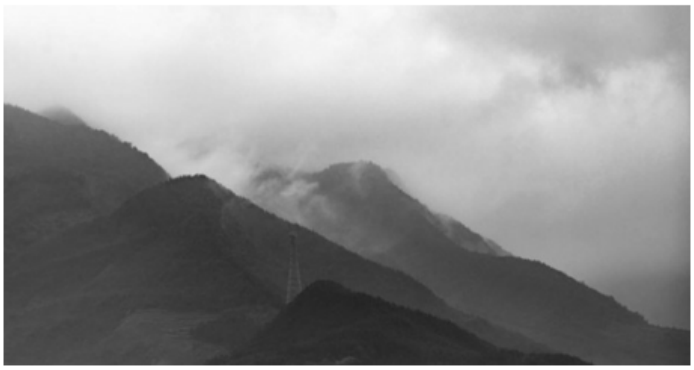
In [13]:
```
### 영상 출력
titles = ["원래 영상", "히스토그램 평활화"]
images = [img_lena_gray, img_hist_equal]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(1, 2, i+1), plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```

원래 영상

히스토그램 평활화

▣ 예제

- 파일
  - pc5.jpg
  - fig 3-7.jpg



▶ 컬러 영상의 히스토그램 평활화

- BGR 영상 → YCC 변환 → Y(grayscale) 히스토그램 평활화 → BGR 변환

In [14]:
```
### 영상 읽기
img_lena = cv2.imread(r'D:WimageWlena.png')
img_lena.shape
```

Out[14]:
```
(512, 512, 3)
```

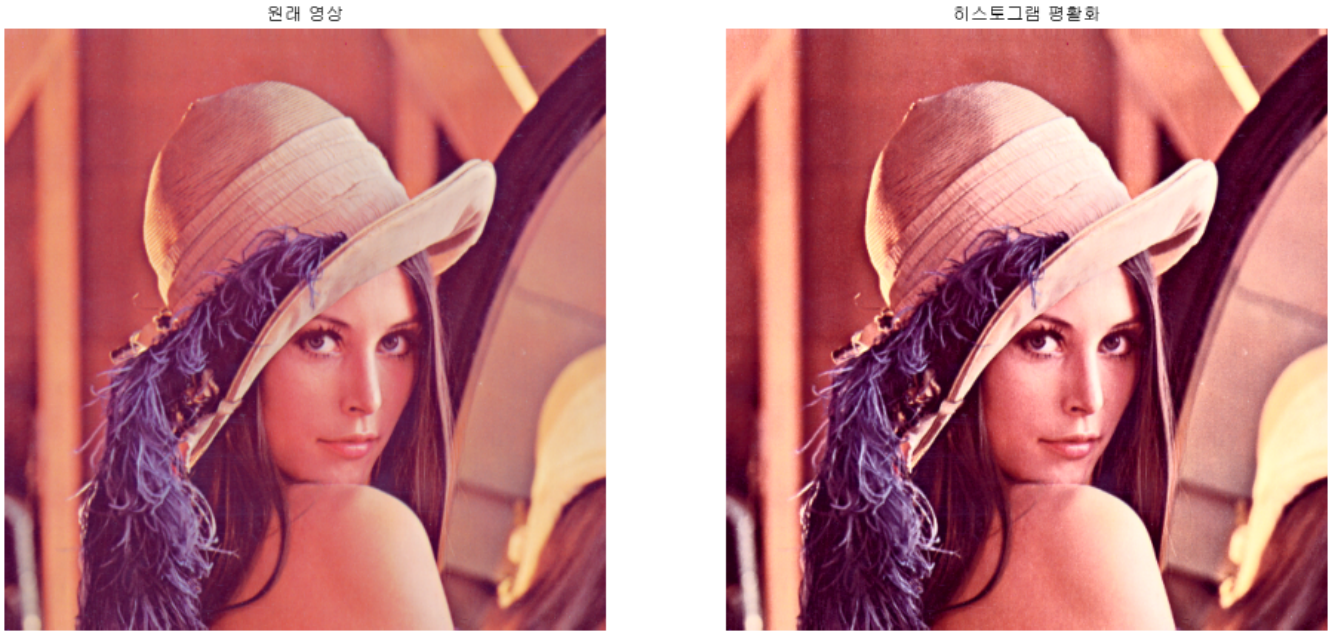
In [15]:
```
### BGR → YCrCb 변환
img_lena_ycc = cv2.cvtColor(img_lena, cv2.COLOR_BGR2YCrCb)
```

In [16]:
```
### Histograms Equalization in OpenCV
img_lena_ycc[:,:,0] = cv2.equalizeHist(img_lena_ycc[:,:,0])
```

In [17]:
```
### YCrCb → BGR 변환
img_bgr_he = cv2.cvtColor(img_lena_ycc, cv2.COLOR_YCrCb2BGR)
```

In [18]:
```
### RGB 영상으로 출력
titles = ["원래 영상", "히스토그램 평활화"]
images = [img_lena, img_bgr_he]
for i in range(len(images)):
```

```
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(1,2,i+1),plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```



▶ 히스토그램 정합(Histogram Matching)

- 개념
  - 영상의 히스토그램이 특정 모양의 히스토그램 분포를 가지도록 하여 영상의 명암 대비를 개선하는 기법
  - 히스토그램 명세화라 불리기도 함
- 과정
  - (1) 입력 영상에 대하여 히스토그램 평활화를 수행
  - (2) 목표 히스토그램에 대하여 히스토그램 평활화를 수행 후 균일 분포 히스토그램 변환 함수를 얻는다.
  - (3) 2의 과정에서 얻은 평활화 된 목표 히스토그램의 변환 함수를 이용하여 역 평활화를 위한 **역변환 함수**를 구한다.
  - (4) 3의 과정에서 얻은 역 변환 함수를 히스토그램 평활화가 적용된 입력 영상에 적용

In [19]:
```
### Package install
#!pip install skimage
```

In [20]:
```
### Package
from skimage import exposure
```

In [21]:
```
### 영상 읽기 - lena.png
img_lena_gray = cv2.imread(r'D:WimageWlena.png', 0)
img_lena_gray.shape
```

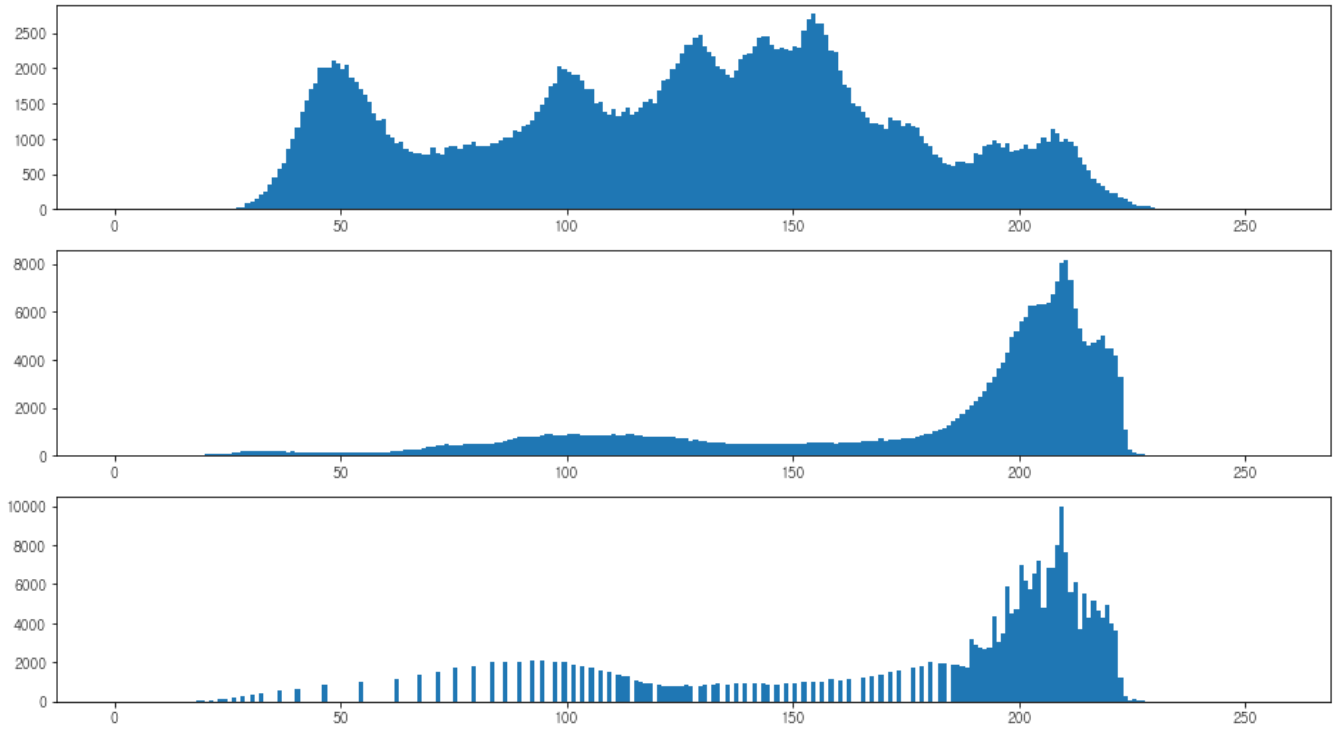
Out[21]:
```
(512, 512)
```

In [22]:
```
### 영상 읽기 - airplane.png
img_air_gray = cv2.imread(r'D:WimageWairplane.png', 0)
img_air_gray.shape
```

Out[22]:
```
(512, 512)
```

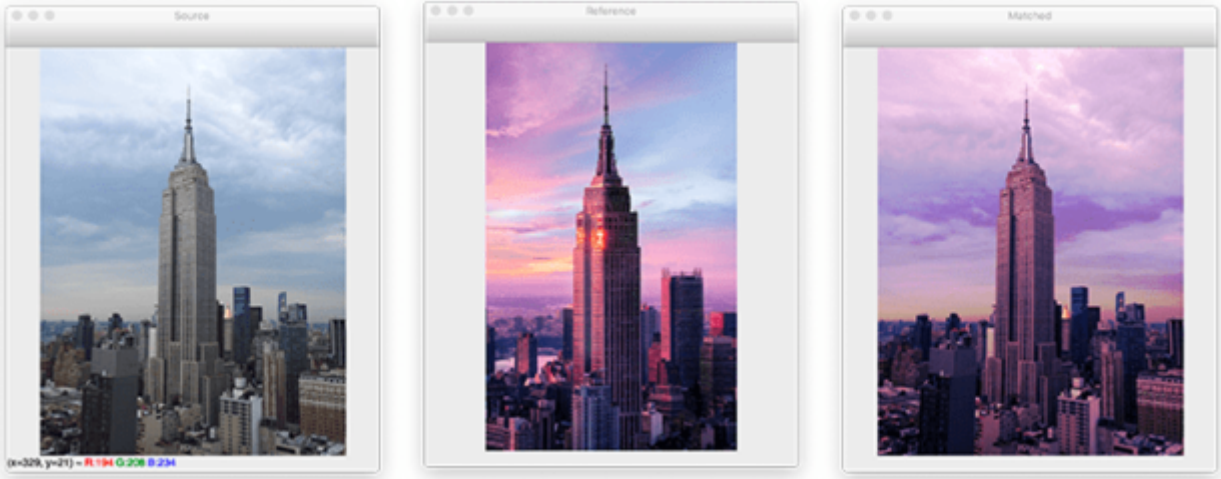
In [23]:
```
### Histogram matching with OpenCV, scikit-image
img_hist_match = exposure.match_histograms(img_lena_gray, img_air_gray).astype('uint8')
```

In [24]:
```
### 히스토그램 출력
plt.subplot(3, 1 , 1), plt.hist(img_lena_gray.flatten(), 256, [0,256])
plt.subplot(3, 1 , 2), plt.hist(img_air_gray.flatten(), 256, [0,256])
plt.subplot(3, 1 , 3), plt.hist(img_hist_match.flatten(), 256, [0,256])
plt.show()
```



In [25]:
```
### 영상 출력
titles = ["lena", "airplane", "Histogram Matching"]
images = [img_lena_gray, img_air_gray, img_hist_match]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(1, 3, i+1), plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```

▶ 컬러 영상의 히스토그램 매칭



- https://pyimagesearch.com/2021/02/08/histogram-matching-with-opencv-scikit-image-and-python/

In [26]:
```
### 영상 읽기
img_lena = cv2.imread(r'D:₩image₩lena_color.png')
img_air = cv2.imread(r'D:₩image₩airplane.png')
```

In [27]:
```
### Histogram matching with OpenCV, scikit-image
img_hist_match = exposure.match_histograms(img_lena, img_air).astype('uint8')
```

In [28]:
```
### 영상 출력
titles = ["lena", "airplane", "Histogram Matching"]
images = [img_lena, img_air, img_hist_match]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(1, 3, i+1), plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```

▣ 예제

- 파일
  - Empire_State_1.jpg
  - Empire_State_2.jpg
  - Empire_State_3.jpg

In [29]:
```
### 영상 읽기
img_ES1 = cv2.imread(r'D:₩image₩Empire_State_1.jpg')
img_ES2 = cv2.imread(r'D:₩image₩Empire_State_3.jpg')
```

In [30]:
```
### Histogram matching with OpenCV, scikit-image
img_hist_match = exposure.match_histograms(img_ES1, img_ES2).astype('uint8')
```

In [31]:
```
### 영상 출력
titles = ["Empire_State_1", "Empire_State_2", "Histogram Matching"]
images = [img_ES1, img_ES2, img_hist_match]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(1, 3, i+1), plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```