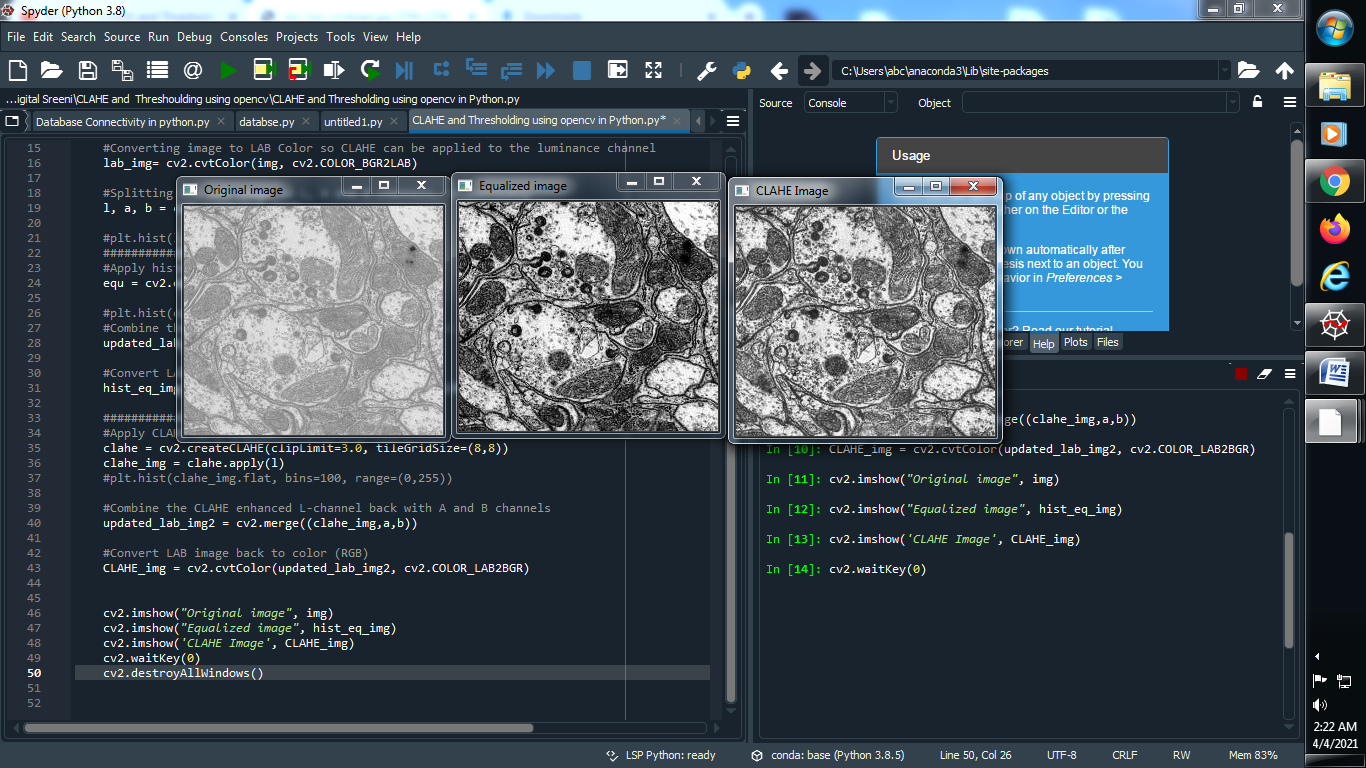
**Histogram Equalization**

|  |
| --- |
|  |
|  | → Stretches histogram to include all ranges if the original histogram is confined |
|  | only to a small region - low contrast images. |
|  | → But, this type of stretching may not result in ideal results and gives |
|  | too bright and too dark regions in the image. This can be very bad for images |
|  | with large intensity variations. |
|  |  |
|  | → CLAHE: COntrast limited adaptive histogram equalization |
|  | Regular histogram equalization uses global contrast of the image. This results in |
|  | too bright and too dark regions as the histogram stretches and is not confined |
|  | to specific region. |
|  |  |
|  | → Adaptive histogram equalization divides the image into small tiles and within |
|  | each tile the histogram is equalized. Tile size is typically 8x8. |
|  | → If the image contains noise, it gets amplified during this process. Therefore, |
|  | contrast limiting is applied to limit the contrast below a specific limit. |
|  | → Bilinear interpolation is performed between tile borders. |
|  |  |
|  | →Below, let us perform both histogram equalization and CLAHE and compare the results. |
|  | → The best way to work with color images is by converting them to luminance space, |
|  | e.g. LAB, and enhancing lumincnace channel only and eventually combining all channels. |
|  |  |
|  | """ |
|  |  |
|  | import cv2 |
|  | from skimage import io |
|  | from matplotlib import pyplot as plt |
|  |  |
|  | img = cv2.imread("images/bio\_low\_contrast.JPG", 1) |
|  | #img = cv2.imread('images/retina.jpg', 1) |
|  |  |
|  | #Converting image to LAB Color so CLAHE can be applied to the luminance channel |
|  | lab\_img= cv2.cvtColor(img, cv2.COLOR\_BGR2LAB) |
|  |  |
|  | #Splitting the LAB image to L, A and B channels, respectively |
|  | l, a, b = cv2.split(lab\_img) |
|  |  |
|  | #plt.hist(l.flat, bins=100, range=(0,255)) |
|  | ###########Histogram Equlization############# |
|  | #Apply histogram equalization to the L channel |
|  | equ = cv2.equalizeHist(l) |
|  |  |
|  | #plt.hist(equ.flat, bins=100, range=(0,255)) |
|  | #Combine the Hist. equalized L-channel back with A and B channels |
|  | updated\_lab\_img1 = cv2.merge((equ,a,b)) |
|  |  |
|  | #Convert LAB image back to color (RGB) |
|  | hist\_eq\_img = cv2.cvtColor(updated\_lab\_img1, cv2.COLOR\_LAB2BGR) |
|  |  |
|  | ###########CLAHE######################### |
|  | #Apply CLAHE to L channel |
|  | clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8)) |
|  | clahe\_img = clahe.apply(l) |
|  | #plt.hist(clahe\_img.flat, bins=100, range=(0,255)) |
|  |  |
|  | #Combine the CLAHE enhanced L-channel back with A and B channels |
|  | updated\_lab\_img2 = cv2.merge((clahe\_img,a,b)) |
|  |  |
|  | #Convert LAB image back to color (RGB) |
|  | CLAHE\_img = cv2.cvtColor(updated\_lab\_img2, cv2.COLOR\_LAB2BGR) |
|  |  |
|  |  |
|  | cv2.imshow("Original image", img) |
|  | cv2.imshow("Equalized image", hist\_eq\_img) |
|  | cv2.imshow('CLAHE Image', CLAHE\_img) |
|  | cv2.waitKey(0) |
|  | cv2.destroyAllWindows() |

****