September 12, 2023

AI6121 – Computer Vision Assignment # 1

Reinelle Jan Bugnot
G2304329L

Part 1.1 Import Data

I began the assignment by loading the target image using the *cv2.imread()* function of the *cv2* python library. I then converted the colored image to grayscale as part of the scope of the lecture.

```python
def load_img(img_path):
    img = cv2.imread(img_path)

    # Convert image to grayscale
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img
```

Fig 1. Code to load image.

Part 1.2 Generate Image Histogram

To generate the image histogram, I initialized a 256-bin array with zeros, flattened the image using *np.ravel()*, and incremented the corresponding histogram bin for each pixel value in a for loop. Afterwards, I calculated the cumulative distribution function (CDF) of the histogram using *np.cumsum()*, and then created a normalized version of the CDF for plotting purposes. The output histogram and CDF plot of a sample image is shown below.

```python
def calc_img_histogram(img):

    # Compute the histogram
    histogram = np.zeros(256)

    # Increment histogram index equivalent to pixel value to create a histogram
    for pixel in img.ravel():
        histogram[pixel] += 1

    # Compute the cumulative sum
    cdf = np.cumsum(histogram)

    # Normalize cdf for plotting purposes within the range of 0 to max histogram
    cdf_normalized_for_plot = cdf * float(histogram.max()) / cdf.max()

    return histogram, cdf_normalized_for_plot
```

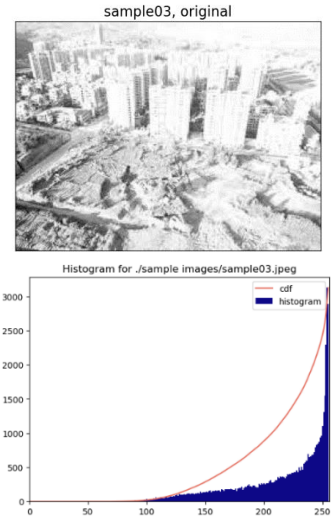

Fig 2. Function to generate histogram (left), sample image (top), and output histogram (bottom).

Part 1.3 Histogram Equalization

Histogram equalization (HE) operates by applying a transformation function to an image, utilizing a normalized Cumulative Distribution Function (CDF) within the 0-255 range to align with 8-bit grayscale image pixel values (as shown in Figure 3) [1]. This allows us to redistribute pixel values across the 0-255 range, enhancing image contrast. The original pixel values are then replaced with equalized values obtained from this normalized CDF by iterating through each image pixel. From these new equalized image pixel values, I can generate equalized histograms and CDFs for plotting, following the same steps as outlined in Part 1.2 of this assignment.

```python
def calc_equalized_histogram(img, histogram):

    # Create an array of zeros with same dimension as image
    img_he= np.zeros_like(img)

    # Calculate cdf of original histogram
    cdf = np.cumsum(histogram)

    # Normalize the cumulative distribution function within the range 0-255
    cdf_normalized = ((cdf - cdf.min()) * 255) / (cdf.max() - cdf.min())

    # Map the original pixel values to the equalized values using the normalized
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            img_he[i, j] = cdf_normalized[img[i, j]]

    # Compute the histogram of the equalized image
    histogram_he= np.zeros(256)
    for pixel in img_he.ravel():
        histogram_he[pixel] += 1

    # Compute the cdf of the new equalized histogram
    cdf_he= np.cumsum(histogram_he)

    # Normalize cdf for plotting purposes within the range of 0 to max histogram
    cdf_he_normalized_for_plot = cdf_he * float(histogram_he.max()) / cdf_he.max

    return histogram_he, cdf_he_normalized_for_plot, img_he
```

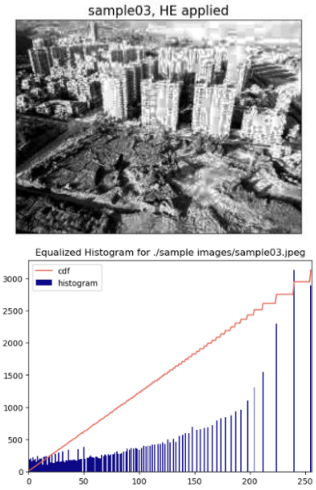

Fig 3. Function to implement histogram equalization (left), equalized image (top), output equalized histogram and CDF (bottom)

As shown in Figure 3, the equalized cumulative distribution function roughly approximates a line, whereas in Figure 2, the CDF function appears as a curve, which exhibits the uneven distribution of pixel values present in the original image. This is because the goal of histogram equalization is to approximate the histogram of the original image into a uniform distribution, which results in a nearly linear CDF.

I can then apply both *calc_img_histogram()* and *calc_equalized_histogram()* functions across all of the 8 provided sample images to observe how histogram equalization interacts with different images, as shown in ***Appendix A***.

Part 2. Pros and Cons of Histogram Equalization

As shown in Part 1, Histogram Equalization is a relatively simple algorithm that can significantly enhance the contrast in an image by stretching the pixel values across the entire range (0-255). This process enhances the details in both dark and bright regions of the image by extending its dynamic range, or the gap between the darkest and brightest pixels.

However, it is important to note that Histogram Equalization is a **global operation**, and thus treats different images the same way. When the original image already possesses a substantial dynamic range (a.k.a., already has areas of high contrast), applying histogram equalization leads to the compression of intensity values in these localized spots which results in loss of information, as demonstrated in Fig 4.

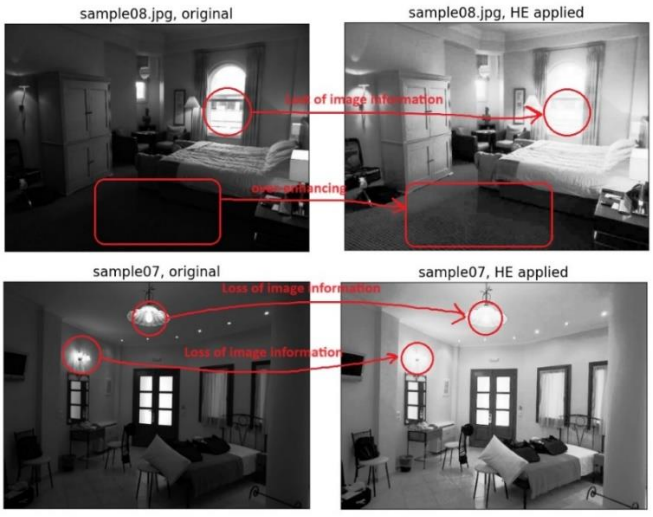

Fig 4. Sample images and their Histogram Equalized counterparts, illustrating some problems with Histogram Equalization.

Part 3. Histogram Equalization Improvements

In order to remedy this, instead of applying the histogram equalization algorithm globally, we can create local patches (or tiles) across the image, and then apply histogram equalization across those localized patches. By doing so, we effectively *reduce* the dynamic range of the image input to HE, which mitigates overcompression of pixel intensity values, preventing loss of information. This algorithm is commonly known as *adaptive histogram equalization* or AHE [1]. We can see this information preservation effect in Fig. 5, where the bulb is completely lost in the global HE, but somewhat preserved in AHE.

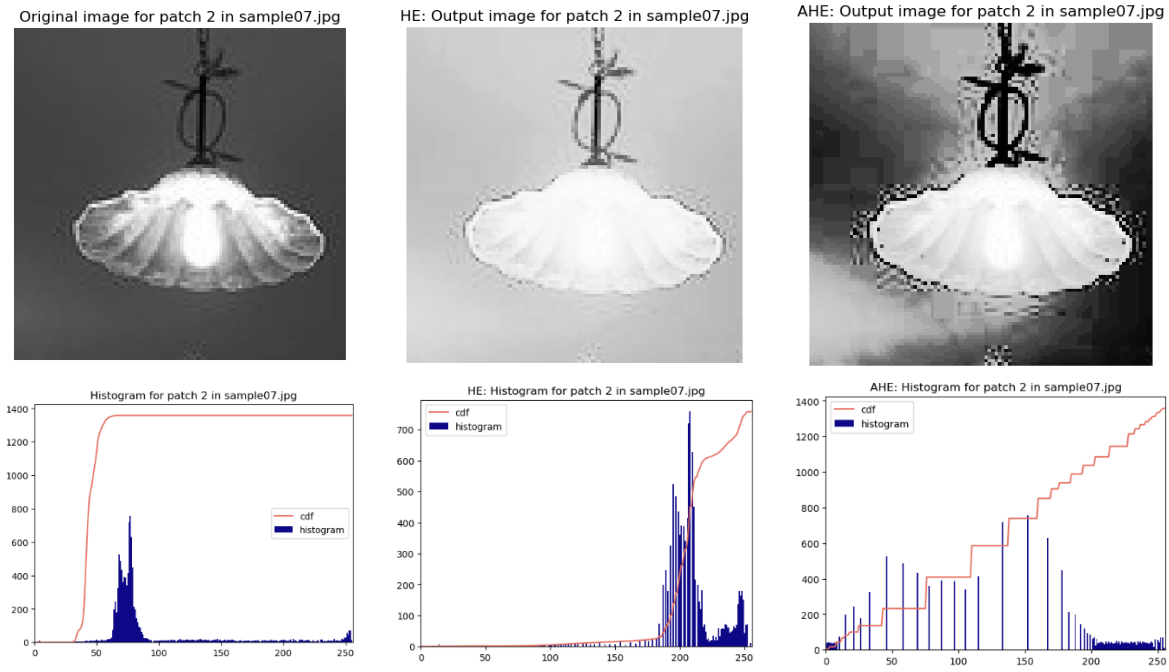

Fig 5. Original image and histogram (left), HE patch output and histogram (center), AHE patch output and histogram (right)

However, since Histogram Equalization (HE) is applied to a very limited subset of pixels, it can not only enhance contrast in localized areas but also undesirably amplify noise [1], as evident in Figure 5. Moreover, localized HE lacks crucial global image context, generating each patch independently and leading to visible contrast boundaries between them. These problems are addressed by a more advanced variant of AHE called Contrast-Limited Adaptive Histogram Equalization (CLAHE), which applies a *clip-limit* to the generated local histograms to mitigate noise overamplification, and *bilinear interpolation* which stitches together the individual patches in a process that smoothens out the contrast boundaries [2].

Fig 6. Adaptive Histogram Equalization output image showing visible contrast boundaries between patches

For the purpose of simple demonstration, I used an already existing function from the openCV library to show how CLAHE improves the contrast of a given input image while mitigating loss of information resulting from over-/underexposure, preventing the amplification of noise, as well as removing the contrast boundaries visible in the basic application of AHE (without interpolation) [3]. The output image is shown in Fig. 7.

```
# For demonstration purposes, I will use an already existing function from the CV2 library
clahe = cv2.createCLAHE(clipLimit=4)
img_clahe = clahe.apply(img)
```

sample07, Contrast Limited Adaptive HE applied

Fig 7. Contrast Limited Adaptive Histogram Equalization. Code that calls function from openCV library (top), output image (bottom).

References:

* ChatGPT, *used to rephrase paragraphs for better grammar and more concise explanations. All ideas in the report belong to me unless otherwise stated*. Chat Reference: https://chat.openai.com/share/404b7e4b-0763-4e50-a969-939dc7cce0b4, 2023

[1] K. S. Htoon, “A tutorial to histogram equalization,” Medium, https://medium.com/@kyawsawhtoon/a-tutorial-to-histogram-equalization-497600f270e2 (accessed Sep. 15, 2023).

[2] “Contrast Limited adaptive histogram equalization,” Contrast Limited Adaptive Histogram Equalization - MATLAB & Simulink, https://www.mathworks.com/help/visionhdl/ug/contrast-adaptive-histogram-equalization.html (accessed Sep. 15, 2023).

[3] “OpenCV: cv::CLAHE Class Reference.” OpenCV documentation index. Accessed: Sep. 15, 2023. [Online]. Available: https://docs.opencv.org/4.x/d6/db6/classcv_1_1CLAHE.html

Appendix A. Histogram Equalization across all 8 sample images