Computer Vision

Subhasish Basak (MDS201803)

Assignment 1

January 29, 2020

The link to my **Colab Notebook** is as follows: Click to view Colab Notebook

(Q.1) Convert a color image into a grayscale image: (3 points)

- We have used the input image **hill.png** and converted it to a grayscale image using the Python package **PIL**. The converted grayscale image is as follows.

Figure 1: hill_gray.png

(Q.2)(a) Read in a grayscale image and linearize the intensity values: (3 + 5 = 8 points)

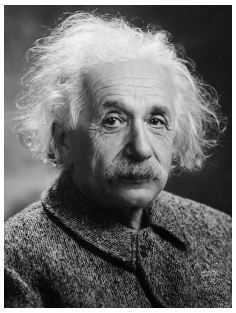

Figure 2: original image

- We load the image in **Einstein.jpg** into a Numpy array. The image has the following characteristics:
  - Dimension of the image : (540, 720)
  - Max : 255
  - Min : 0
  - Image format : JPEG
- As we can see the integer values range from 0 to 255, i.e. a total of $256 = 2^8$. Thus there are 8 bits per integer are there in the image.

- The Height and Width of the image is 720 **pixels** and 540 **pixels** respectively.

(Q.2)(b) Linearizing the image

- Converting the image into an array within the range $[0, 1]$.This linear normalization of the grayscale digital image is performed according to the formula..

$$I_N = (I - \text{Min})\frac{\text{NewMax} - \text{NewMin}}{\text{Max} - \text{Min}} + \text{NewMin} \tag{1}$$

  Here,

  - I_N = Pixle values of the Normalized Image
  - Min, Max = Intensity bounds of the old image
  - NewMin, NewMax = Intensity bounds of the Normalized image

(Q.2)(c) Observations

- The image after linearization is as follows. We do not notice any visible change to the image.

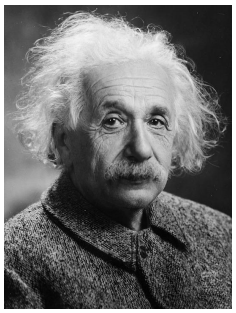

Figure 3: linearized image

(Q.3) Formulas for computing the padded pixel values.
Suppose $f(k, l)$ be the $M \times N$ dimesional image pixles and $f_p(i, j)$ be the pixles of the padded image. Let us pad the image M_1 and M_2 units in the top and bottom respectively and also N_1 and N_2 units in the left and right respectively.

- **Constant Padding**

$$f_p(i,j) = \begin{cases} f(k,l) & k = i - M_1, l = j - N_1 \text{ and } 0 \le k \le M, 0 \le l \le N \\ k & \text{otherwise} \end{cases} \tag{2}$$

  where, $0 \le i \le M + M_1 + M_2, 0 \le j \le N + N_1 + N_2$

- **Zero Padding**

$$f_p(i,j) = \begin{cases} f(k,l) & k = i - M_1, l = j - N_1 \text{ and } 0 \le k \le M, 0 \le l \le N \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

  where, $0 \le i \le M + M_1 + M_2, 0 \le j \le N + N_1 + N_2$

- **Clamp Padding**

$$f_p(i,j) = f(k,l) \tag{4}$$

  where, $k = max(0, min(M - 1, i))$ & $l = max(0, min(N - 1, j))$

- **Mirror Padding**

$$f_p(i,j) = f(k,l) \tag{5}$$

  where, $k = max(0, min(M - i, i))$ & $l = max(0, min(N - j, j))$

(Q.4) Padding images
We use the padding modes on different images. Here are some of the images. Rest of the images are shown in my **Jupyter Notebook**.

(a) Zero padding

(b) Mirror padding

(c) Clamp padding

(d) Constant padding

Figure 4: Padding Examples

(Q.5) Convolution

- Gaussian filter with a 5×5 kernel given by,

$$\frac{1}{256} \times \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

(a) Original

(b) Filtered

- Box filter with a 5×5 kernel given by,

$$\frac{1}{49} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

(a) Original

(b) Blurred

(Q.6) Image Sharpening

- We have used both the Gaussian filter and the box filter to blur the image first and then the blurred images are subtracted from the doubled intensity original image to build the sharpen images.
  - Sharpening after Gaussian & box filtering

(a) Original

(b) Sharpned Gaussian

(c) Sharpned Box

The kernels used are same as the one mentioned in **(Q.5)**

(Q.7) Sharpening using Median filter

- The median filter replaces the pixle value with the median computed out of its neighbour. Here is an example with a 7×7 median filter.

(a) Original

(b) Filtered

(Q.8) Comparing the filters
In the above questions **(Q.6)** and **(Q.7)** we have used the *Gaussian* filter and the *Box* filter to smooth out the images and then sharpen them.
For Blurring effect the **Box** filter performed better then the **Gaussian** filter. Also as the kernel size increases the Blurring as well as smoothing effect is more prominent.
For sharpening the images we have used the Median filter. Here also as the kernel size increases the sharpening effect is more prominent.