Computer Vision - Assignment 1

Rohan Khaitan - MDS201812

Jan 2020

Link of notebook : `https://colab.research.google.com/drive/1fz2gLpr3zD2cRLNaRjVTuELDdbs8MYfp`

Question 1:

Convert a color image into a grayscale image: (3 points) Read the input image "hill.png". Convert it to a grayscale image, display and save it as "hill_gray.png"

Answer:

The image "hill_png is converted to gray scale image.

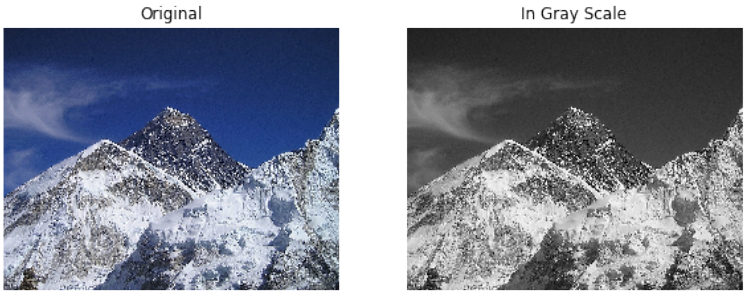

Figure 1: Original and gray scale image

Question 2:

Read in a grayscale image and linearize the intensity values: (3 + 5 = 8 points) a) Load the image in "Einstein.jpg" into a Numpy array. Originally, it will be in the form of a 2D-array of unsigned integers. Check and report how many bits per integer there are in the image, and what its width and height is. Then, convert the image into a double-precision array. b) Convert the image into an array within the range [0,1]. Do this by applying a linear transform (shift and scale) to the image, so that the minimum intensity value is mapped to 0, and the maximum intensity value is mapped to 1. Now, multiply the intensity values by the maximum value for the representation (eg. 255 for 8-bits). Display and save the image after linearization. c) What do you observe in the linearized image?

Answer:

(a) The data type of the Einstein image is obtained as uni8 i.e.unsigned integer number stored with 8 bit. So the bits per integer in the image is 8.
Height of the image: 720 Pixels.
Width of the image: 540 Pixels.
The image is converted to double precision array. The output array is shown in the notebook.

(b) Minimum Value of pixel in the image is 0 and maximum value of pixel is 255. The image is converted to an array within the range [0,1] by subtracting minimum pixel value from the pixel values and then by dividing it by maximum pixel value.
After multiplying 255 with the converted array, the obtained image stays the same as the original one.
(c) The linearized image stays the same as the original image. This is shown in the notebook.

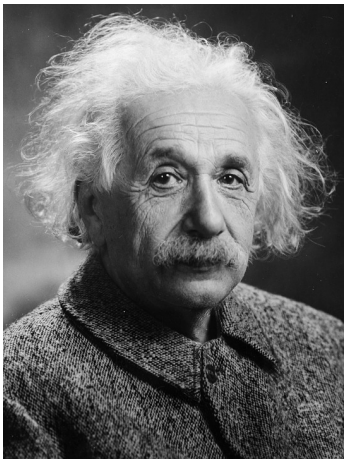

Figure 2: Original and transformed image stays same

Question 3:

(8 points) Write down the formulas for computing the padded pixel values f ” (i, j) as a function of the original pixel values f(k, l) and the image width and height (M, N) for each of the padding modes:

- Zero
- Constant
- Clamp
- Mirror

Answer:

For a coloured image we have 3 channels -R, G, B. For each channel we will have a matrix. For a gray scale image we only have one matrix.
Let f be the M x N image matrix. Let g be the padded image of size M_2 x N_2.

Formula for Zero Padding:

For $1 <= i <= M_2$ and $1 <= j <= N_2$,
$g(i,j) = f(i,j) \quad if \quad (M_2 - M)/2 + 1 <= i <= M + (M_2 - M)/2$ and $(N_2 - N)/2 + 1 <= j <= N + (N_2 - N)/2$
$g(i,j) = 0 \quad otherwise$

Formula for Constant Padding:

For $1 <= i <= M_2$ and $1 <= j <= N_2$,
$g(i,j) = f(i,j) \quad if \quad (M_2-M)/2+1 <= i <= M+(M_2-M)/2$ and $(N_2-N)/2+1 <= j <= N+(N_2-N)/2$
$g(i,j) = k \quad otherwise \quad (k - constant)$

Formula for Clamp Padding:

$g(i,j) = f(k,l) \qquad k = max(0, min(M-1, i)), l = max(0, min(N-1, j))$

Formula for Mirror Padding:

$g(i,j) = f(|i|, |j|) \qquad if \quad i <= M-1, j <= N-1$
$g(i,j) = f(2M-i-1, |j|) \qquad if \quad i > M-1, j <= N-1$
$g(i,j) = f(|i|, 2N-j-1) \qquad if \quad i <= M-1, j > N-1$
$g(i,j) = f(2M-i-1, 2N-j-1) \qquad if \quad i > M-1, j > N-1$

Question 4:

(16 points) Write a function that will take as input an image (grayscale or color), a padding specification and width of padding in each dimension and returns the appropriately padded image. Test your function on the given input images for widths of 1, 2 and 3 in each dimension. Display and store your results.

Answer:

The padding function is applied on different images. Different width of padding is used for different images. Following images showing result for each type of padding -

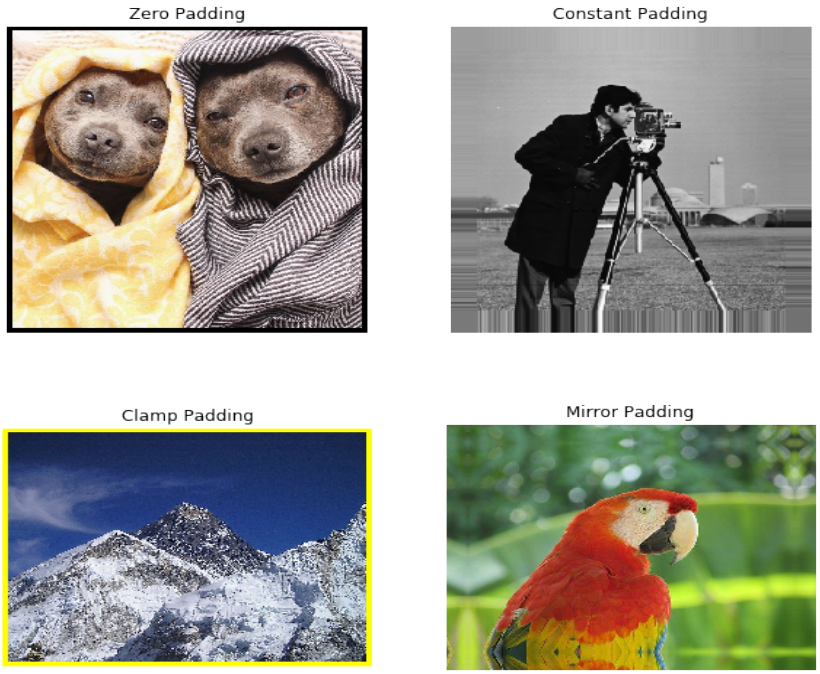

Figure 3: Different padding in different images

Question 5:

(25 points) Implement convolution with a separable kernel. The input should be an image (grayscale or color), along with the vertical and horizontal kernels. Utilize the padding functions implemented above. Realize the box filter and the Gaussian filter for different kernel sizes (5x5, 7x7, 11x11) using your convolution function.

Answer:

The convolution function is written in the notebook. Box filters and Gaussian filters for different kernel sizes (5x5, 7x7, 11x11) are shown. The function also accepts kernel as input if we set generate_kernel argument of function to false. Default is true and it generates kernel automatically based on input of the kernel size and other parameters(sigma in case of Gaussian). The generate_kernel function generates the vertical and horizontal kernels. The generate_kernel is written only to get input for the function easily. We can provide our own vertical and horizontal kernels also. Images after applying the filters are shown later in the next answer.

Question 6:

(10 points) Apply the box and Gaussian filters of different sizes to smooth, and then sharpen the given input images. Store your results.

Answer:

Smoothing: The results of using Box and Gaussian kernel of different sizes on the two images are given below -

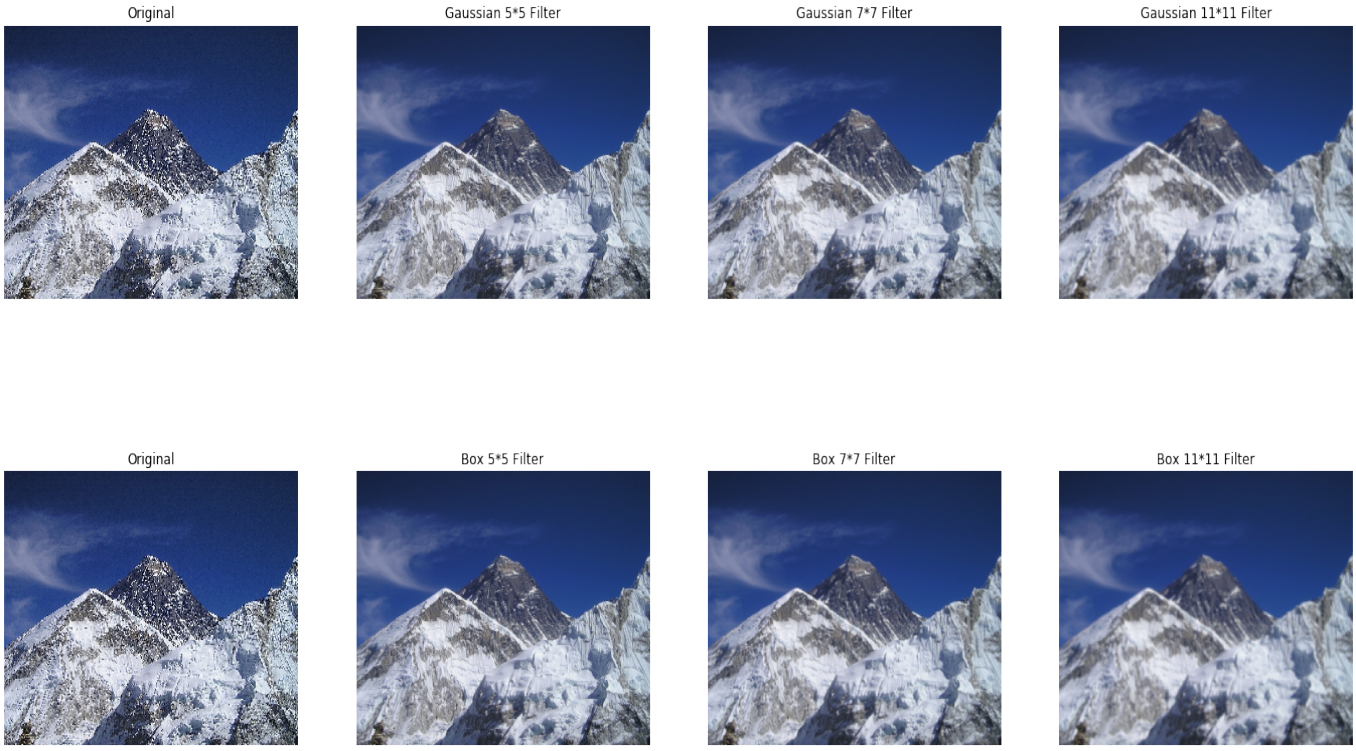

Figure 4: Smoothing using different filters

Figure 5: Smoothing using different filters

Sharpening: The Box and Gaussian Filter can be used to sharpen the image. The sharpening is done by adding the original image twice(or maybe by an constant k) and subtracting blurred version of that image from that. The following formula is used for sharpening :

g(x,y) = f(x,y) + $f_{smooth}(x, y)$
$f_{sharp}(x, y) = f(x, y) + k * g(x, y)$

Figure 6: Sharpening using 5*5 Box and Gaussian Filter

We got different blurred images using box and gaussian filters of different sizes. Here I just showed two sharpened images obtained using 5*5 Box and Gaussian filter. In the last question other example is shown using different sized filters.

Question 7:

(20 points) Implement the median filter using the same filter sizes as with the Gaussian and box filters. As with the linear filters above, apply the median filters of different sizes to smooth, and then sharpen the given input images. Test and store your results.

Answer:

Separate convolution function is written to apply convolve filter on the image. The concept is same, instead of element wise sum, median of the elements is taken. Median filter helps to remove the noise from the image. Median filtering is one kind of smoothing technique. Here effect on two image is shown. The smoothing effect is clear if we see the transformed images. It removed the noise from the image pretty well.

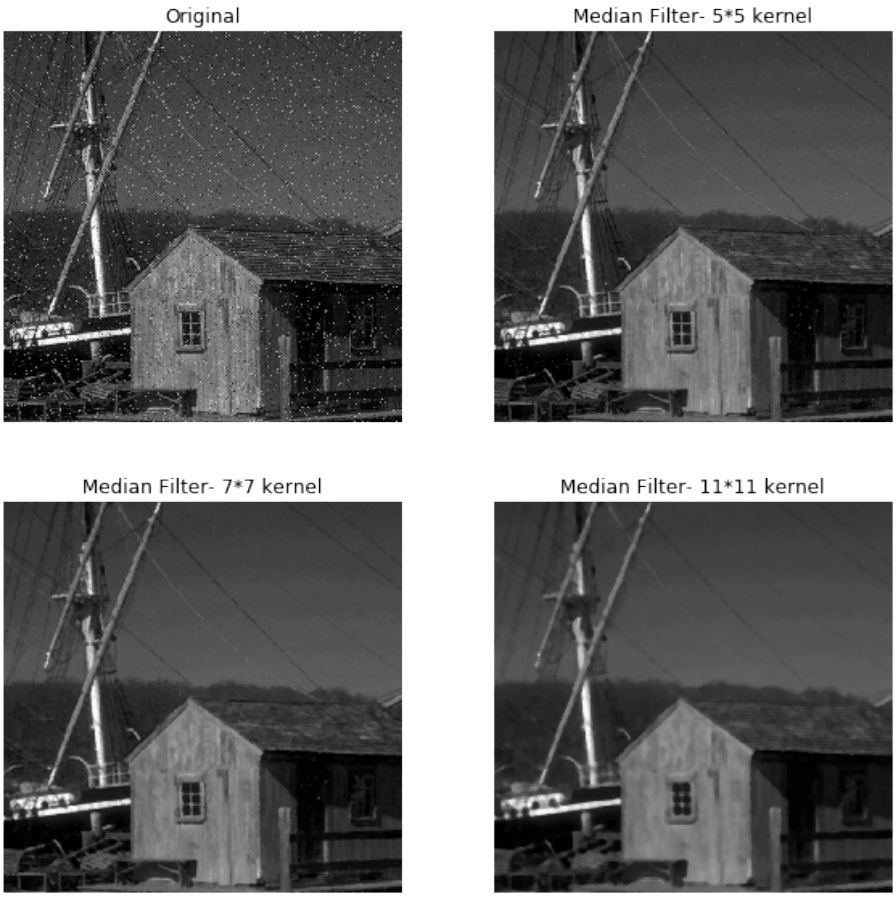

Figure 7: Median filter of different filter sizes applied on a noisy image

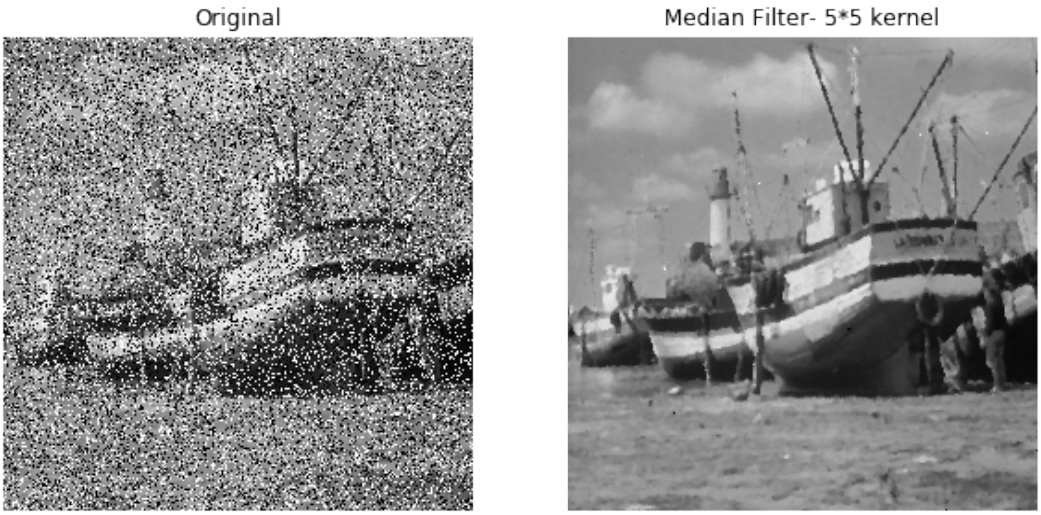

Figure 8: Effect of a 5*5 median filter on a noisy image

Question 8:

(10 points) Compare your results from tasks 6 and 7 above and identify the filters that provide the best results for smoothing and sharpening on the given input images.

Answer:

The following formula is used for sharpening :

$g(x,y) = f(x,y) + f_{smooth}(x,y)$
$f_{sharp}(x,y) = f(x,y) + k * g(x,y)$

Figure 9: Image Sharpening using Box, Gaussian and Median filters

Though it seems that all the sharpened images obtained using Box, Gaussian and Median filters of different sizes, if we look closely we observe that Gaussian Filter worked better for sharpening in case of this particular image. Gaussian 11*11 filter in particular did well to sharpen. For other cases also we observe the sharpening effect. The sharpening effect also depends on the image on which we are applying the filters. For some of the image we won't be able to distinguish the effect at all. Sharpening effect also depends on the constant k (in the formula). The comparison is shown in the notebook also for some images.
For smoothing box filter worked better than others for the images I chose. Gaussian worked better than the median filter. Median filter mainly helped in removing the noises. The smoothing effect for the images is shown earlier in the pdf and also shown in notebook.