EEE424 - Mini Project

Introduction to Digital Image Processing

(Due 21 May 2023)

Typically, an analog optical image is formed by optics (lenses, propagation of light, and possible optical analog processing in that path) on the image capturing element of the camera. The image capturing element is usually a planar pixelated device, that can capture a 2D digital image; the captured image may be black-and-white (a single 2D array) or color image (typically three 2D arrays, where each array corresponds to basic colors red, green and blue (RGB)). A black-and-white digital image consists of pixels, and it is mathematically represented as a 2D array, which is just a matrix.

The captured 2D array by the image capturing element of the camera is called "raw image" which means no further processing is applied. Usually further processing is immediately applied by commonly used digital consumer cameras to enhance the artistic appeal of the original (raw) image, and to compress the number of bits needed to represent the data. The adopted compression algorithms almost always comply with some standard compression procedures (for example JPEG compression) and they are usually lossy compression algorithms. As a consequence of lossy compression, the image quality degrades, but storage and transmission of the image gets easier due to smaller file size.

You will find a black-and-white picture uploaded to MOODLE for this project. It is a black-and-white image whose size is 4096×4096 pixels, and each pixel is represented by 8 bits. Therefore, this is a high resolution image. You may note that, standard-definition TV is 480×640 pixels, high-definition TV is 720×1280, 2K digital TV is 1080×1920, and ultra-high TV (4K TV) is 2160×3840 pixels; each has 8-bit pixel size for each color.

General Remarks:

* Be aware of the MATLAB array indexing convention: indices in MATLAB arrays start from 1. Note that our finite size arrays in signal processing starts from 0 when defining arrays related to DFTs.

* MATLAB has two different commands for FFT: one of them matches our DFT definition that is also widely used in signal processing; the other one may be more convenient for some graphic displays. Make sure you use the correct one.

* Make sure that all your image outputs (not the intermediate DFT computations) are 8-bits integer type data per pixel.

* Label your output images by the section numbers (example: 1-a-i . If there are many images in that section, use a subsequent index: 1-a-i-1). Use tif image types for your stored images (example: 1-a-i.tif); that will not use compression, and therefore, will not generate additional compression related degradations.

* Upload your generated images, as requested, to MOODLE.

* Generate a report which will fully describe what you did with mathematical clarity with no ambiguities. Your report should not contain unnecessary general content, but focus only on what you did, and how you did it, together with needed results and comments. Include some images in your report, but only those that are needed and whose sizes are appropriate for printing on A4 size paper. Note that every display device, and every printer further distorts the gray level images as a consequence of their capabilities.

* As required for any academic writing, make sure all citations are properly marked and a reference list is included. Make sure you credit the photography artist who took and granted permission to use his/her original picture. Include the appropriate copyright notice on behalf of the artist, and indicate his/her permission. The original picture is downloaded from www.pexels.com; note the license info at www.pexels.com/license. The artist is Todd Trapani.

* Include your MATLAB code as an appendix to the report.

* It is usually desirable to use the full gray-level range (full contrast) when an image is displayed. Design a simple gray-level stretcher to use the full 8-bit gray-level. A pixel which is absolutely dark (black) has a value 0, whereas a brightest pixel (white) has a value $2^8 - 1$ when using 8 bits. However, actual pixel values in an image may not use this full-range of possible 256 levels of different gray levels. Therefore, it is desirable to increase the contrast before displaying an image. To achieve that, find the minimum value x_{min} (may not be 0 in a given image) and the maximum value x_{max} (may not be $2^8 - 1$ in the given image) for a given image. Then, design a line relationship (find a and b) $y = ax + b$ where x is the value of a pixel of the original image in $[x_{min}, x_{max}]$, and y is the value of the corresponding output image pixel whose range is in $[0 - 255]$. Note that an error here will either result in saturated areas which is a severe distortion, or under-utilization of the available dynamic range. Also note that image pixels must be nonnegative, since brightness cannot be negative. Store the contrast stretched image in a file, as a simple matrix whose entries are single byte integer type data; compare the file sizes of the stored original image and stored output image matrix. Note that, original image is in TIFF image format, which does not employ any compression, but has some headers and trailers to comply with the standard. MATLAB can read such standard format images to a matrix.

* A 2D function $f(x, y)$ may be decomposed into a superposition of 2D complex sinusoid functions as:

$$f(x,y) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u,v) e^{j(ux+vy)} du dv \quad .$$

The coefficients $F(u, v)$ which are called the 2D Fourier transform of $f(x, y)$, and can be found as,

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{-j(ux+vy)} dx dy \quad .$$

Here, u and v are spatial frequencies whose units are radians per unit length.

Similar to the 1D case, 2D forward DFT is defined as,

$$X(k_1,k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1,n_2)e^{-j(\frac{2\pi}{N_1}k_1n_1+\frac{2\pi}{N_2}k_2n_2)} \quad ,$$

where, $k_1 = 0,\cdots,N_1-1$, $k_2 = 0,\cdots,N_2-1$. The forward 2D DFT is,

$$x(n_1,n_2) = \frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} X(k_1,k_2)e^{j(\frac{2\pi}{N_1}k_1n_1+\frac{2\pi}{N_2}k_2n_2)} \quad .$$

1- Preliminary Work:

* Study the concept of spatial frequency both in continuous and discrete (normalized) case.

* Try to understand the continuous 2D basis functions, $e^{j(ux+vy)}$. Generate the real parts of their discrete versions $Re\left\{e^{j(\frac{2\pi}{N})(k_1n_1+k_2n_2)}\right\}$, for $N_1 = N_2 = N = 256$ for five different (k_1,k_2) pairs, choosing $k_1,k_2 = 0,\cdots,N-1$; chose rather smaller k's. What you will get are five $N\times N$ matrices. Print those matrices in your report as 8-bit/pixel gray-level images using the gray-level adjustment that utilizes the full dynamic range as you did before; make sure you indicate the corresponding (k_1,k_2) indices. Upload the images also to MOODLE. Comment on the appearance of these basis functions. Comment separately on the horizontal spatial frequency, k_1 and k_2 and their effect on the appearance.

* Learn how to efficiently implement 2D DFT using 1D FFTs. How many 1D FFTs are needed? What will be the size of those FFTs?

2-Size Reduction

a) It is quite common to use different sizes of an image. A naive approach is to directly downsample the image both horizontally and vertically using a rectangular sampling grid:

$$I_1(m,n) = I(Mm,Mn)$$

where $I(m,n)$ is the original version, and $I_1(m,n)$ is its smaller size version. Generate and store a sequence of images by reducing the size of the original image iteratively by 4 times in each directions at a time; i.e., $M=4$, and

$$I_{i+1}(m,n) = I_i(Mm,Mn) \quad i = 1,2,3 \quad .$$

Thus, including the original, you will have a total of four images at different sizes where the smallest one will be 64×64. Upload your images to MOODLE.

Assuming that the original sampling rate to get the large size image when the picture was taken originally (analog-to-digital conversion stage) was adequate to satisfy the Nyquist rate, it is quite likely that your resampling, which corresponds to lower rate sampling of the original analog image, will create aliasing. Carefully examine the image to observe and point out the aliasing related artifacts.

Now, do the downsampling properly, by avoiding aliasing:

b) Design an ideal low-pass filter in the DFT domain by,

i- Taking and saving the 2D $N \times N$ DFT of the large size image,

ii- Convert the DFT coefficients corresponding to higher frequency components of the original image to 0 (do this for approximately 15/16 of the $N \times N$ original DFT coefficients) and keep the coefficients of the rest of the DFT corresponding to low-frequency components unchanged. Make sure you comply with specific details as described in the *Notes* at the end of this section.

iii- Take the $N \times N$ IDFT of the result in (b),

iv- Convert your images to 8-bit/pixel images that utilizes the full 256-level gray dynamic range.

iv- Then apply the downsampling described above.

Repeat the procedure above to go down to 64×64 image. Since your starting size was the original size (4096×4096), you will have three more images (1024×1024, 256×256, and 64×64); store those three images and carefully examine them, and compare the results of unfiltered and low-pass filtered downsampling. Upload your images to MOODLE.

Notes:

Your original image is naturally real valued. You must generate real valued output images, as well. Therefore, your filter impulse response must also be real valued. That imposes symmetry conditions on the filter frequency response: $H(k)$ must be conjugate symmetric. Since your ideal filter frequency response is also real valued (it has either 1 or 0 values), that means even symmetric frequency response. Note that, since we are implementing DFTs, that means circular even symmetry:

$$H(k_1, k_2) = H((-k_1)_{mod\ N}, (-k_2)_{mod\ N}) \quad .$$

An easy (but not necessary) way to achieve this is to have a separable function as the frequency response:

$$H(k_1, k_2) = H_1(k_1)H_2(k_2)$$

where $H_1(k_1)$ and $H_2(k_2)$ are 1D circularly even symmetric functions, $H_1(k_1) = H_1((-k_1)_{mod\ N})$, and, $H_2(k_2) = H_2((-k_2)_{mod\ N})$. For the ideal LPF above, choose,

$$H_1(k) = H_2(k) = 1 \quad \text{if} \quad k = 0, \cdots, N/8 - 1 \quad \text{or} \quad k = N - (N/8 - 1), \cdots, N - 1$$

Note that your DFT domain ideal filter corresponds to circular convolutions in spatial domain. This may not be desirable due to implicit periodic input image that goes through filtering. Effects may be visible at the edges, especially when the periodic extensions of the original image results in sharp boundary mismatches in gray level values. Try to observe and point out those effects in your results (better visible after the filtering, but before the downsampling).

3- Approximations

Start with the saved DFT coefficients of the original image. Take the magnitude of the DFT coefficients.

a) Print out the largest 20 DFT coefficients as a table in your report that indicates both the (k_1, k_2) indices and the corresponding $|X(k_1, k_2)|$.

b) Generate a 4096×4096 image through the following steps:

   i- Determine a threshold value, and let the DFT coefficients lower than that threshold to be 0. Adjust the threshold to keep about 1/4 of the DFT coefficients as they are (above the threshold) and set the others (about 3/4 of them) to 0. (Note that numerical noise built-up may result in DFT coefficients that may not be conjugate symmetric after this step. Check this situation, and restore the symmetry, if necessary.)

   ii- Take the 4096×4096 IDFT.

iii- Convert the result to an 8-bit 4096×4096 image that utilizes the full gray level dynamic range. Store your image. Upload it to MOODLE.

Repeat steps (i) through (iii) for approximately 1/16, 1/64 and 1/256 non-zero DFT coefficients.

Comment on the results of your successive approximations in terms of achieved savings and related quality degradations.