INTRODUCTION TO DIGITAL IMAGE PROCESSING
361.1.4751

EXERCISE 3 - Transformations

Submission Date: 10.2.2024

For any question regarding this assignment please refer to the course forum on the moodle web site, for personal questions **only** please email schorya@post.bgu.ac.il.

1 2D-Fourier Transform

In this section we will assignment the 2D Fourier Transform on images and learn about some of its properties.

1.1 Writing your own functions

In this section you **should NOT** use the following MATLAB functions: *fft(), ifft(), fft2(), ifft2(), fftn(), ifftn(), ifftw() fftshift(), ifftshift().*

1. Write your own 2D-FFT function named *dip_fft2(I)* **and** an inverse-FFT function called *dip_ifft2(FFT)*.

   The equations for the FFT and iFFT for an image I of size $M \times N$:

$$F(u+1, v+1) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m+1, n+1) \cdot e^{-2\pi i\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

$$I(m+1, n+1) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u+1, v+1) \cdot e^{2\pi i\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

   Notes:
   1. Use matrix implementation.
   2. Remember that the input could be complex both for the fft() and the ifft().

2. When analyzing the frequency components of signals, it can be helpful to shift the zero-frequency components to the center. Write your own shift zero-frequency component to center of spectrum along two dimensions, function named *dip_fftshift(FFT)*. This operation is illustrated in Figure 1.

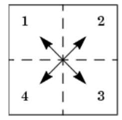

Figure 1: *dip_fftshift(FFT)*

3. Read the *beatles.png* image accompanied to this assignment, and convert it to grayscale normalized image.

4. Compute the 2D-FFT of the image and shift the output image using *dip_fftshift(FFT)* function. Display the log of the amplitude and the pahse of the resulting image. Use *imagesc(-)* and *colorbar* functions to display the results.

5. Reconstruct the original image by using your inverse-FFT function. Is it identical to the original image? Note that the output of the iFFT are complex numbers - you should display only the real part of the image using *imshow(real(-))*.

1.2 Transformation properties

In this section you **may** use the built-in Matlab *fft2(), ifft2() and fftshift()* functions.

1. **Linearity (Free Willy)**

   (a) Load the *freewilly.mat* file enclosed to the assignment. Display it as an image using *imshow()*. **Don't** normalize this file.

   (b) As you can see, Willy the whale is imprisoned. Your job is to free Willy! To do that, you are told that the prison bars are made of a sinudoidal signal in the X axis that was **added** to the original image: $0.5 \cdot sin\left(\frac{2\pi f_x}{N}x\right)$, where N is the number columns in the image. Given this image, find the spatial frequency of the prison bars f_x. Explain your answer! (*Hint:* You may also plot the first row of the image if you find it helpful).

   (c) Based on your answer in section (b), create the image of the prison bars and display it. MAKE SURE it has the same dimensions as *frewilly.mat.*

   (d) Compute the 2D-FFT of the prison bars image, and display its **amplitude** (use the function *abs()*). Explain this result - give a mathematical proof that this is the Fourier transform that is expected here.

   (e) Explain how can you free Willy (i.e. filter out the prison bars) through the Fourier domain. Based on your answer, write a function *Free_Willy(Willy)* that **returns and displays** Willy without the prison bars.

2. **Scaling, translation and seperability**

   (a) Initialize a 128×128 all-zeros matrix. At the center of it, place a 40×40 all-ones square (in pixels 44:83 in each dimension). Display the image and its 2D-FFT. Explain why it looks the way it does.

(b) Initialize a 128×128 all-zeros matrix. At rows 64:103 and columns 64:103, place a 40×40 all-ones square. Display the new image and its 2D-FFT. Does the FFT looks the same as in section (a)? Now Display only the FFT **amplitude** of the previuos and the new image. Are they they identical? Explain why. Base your answer on the **mathematical relationship** between the two images and their 2D-Fourier transforms.

(c) Initialize a 128×128 all-zeros matrix. At the center of it, place a 80×20 all-ones rectangle. Display the new image and its 2D-FFT. Does the FFT looks the same as in section (a)? Explain why. Base your answer on the **mathematical relationship** between the two images and their 2D-Fourier transforms.

(d) Can you represent the image from section (c) using two 1D vectors? Explain how.

(e) Explain how can you compute the 2D-FFT of an image using 1D-FFTs if the image is separable into two 1D vectors. Write a function *sep_fft2(v1,v2)* that receives a pair of 1D vectors (of lengths N1 and N2 respectively), and returns the 2D-FFT (a N1*N2 matrix) based on the 1D-FFTs of the vectors (DO NOT use *fft2()* here). Apply this function on the two vectors you described in section (d), and display the resulting 2D-FFT. Is it identical to the 2D-FFT of the image from section (c)?