INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 3 - Transformations

Submission Date: 1.12.2021

Introduction

In this assignment we will practice in Fourier, Discrete Cosine Transform and Wavelets transformations on the image domain. Although MATLAB has many built in functions, our goal is to learn image processing through doing things ourselves. Please avoid using built-in MATLAB functions unless clearly stated otherwise. However, feel free to compare your own functions to the built-in functions. **Before you start**, please read through the submission instructions at the end of the assignment and follow them during your work. For any question regarding this assignment please refer to the course forum on the moodle web site, for personal questions **only** please email shaked0@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.

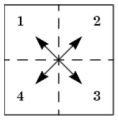

Figure 1: *dip_fftshift(FFT)*

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)?

2 Discrete Cosine Transform

In this assignment we will learn the tools for 2D Discrete Cosine Transform (DCT).

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

2. Use the MATLAB function *dct2(Img)* on the image.
   Display the result using *imagesc(log(abs(DCT)));colormap(jet(64));colorbar;*

3. Randomly, set to zero 50% of the DCT values. Apply the inverse DCT (IDCT) using *idct2(DCT)* MATLAB function on the result. Display the obtained result from the IDCT.

4. Set to zero the DCT 50% absolute lowest values and apply the IDCT on the result. Display the obtained result from the IDCT.

5. Set to zero the DCT values in the range $(-a, a)$. Find the maximum a which will allow you to reconstruct the image using IDCT in a good quality (according to your interpretation).
   What is the precentage of the DCT values that set to zero?
   Display the obtained result from the IDCT.

6. Explain the obtained results from sections $3-5$. In your explanation, consider the meaning of the DCT in compressing image.

3 Wavelet Transform

In this assignment we will learn the tools for 2D Wavelet Decompostion.

1. Read the *beetle.jpg* image (enclosed to this assignment), convert it into grayscale image and normalize to $[0, 1]$.
2. Extract the wavelet decomposition using MATLAB's *wavedec2()* function: use the *'haar'* wavelet, you may choose any level of decomposition between 3-5.
3. Use Matlab's *detcoef2* and *appcoef2* functions to find the detail and approximation coefficients for each level.
4. Display the results for each level.
5. Explain the results. What are the differences between the detail and the approximation coefficients? What are the differences between the horizontal, vertical, or diagonal detail coefficients? What are the differences between each level?

Submission Instructions

The following instructions are mandatory and will be checked and graded by the course staff. Failing to follow these instructions **will** reduce points from you grade.

The assignment is to be done in MATLAB, preferably with MATLAB notebook. **Note:** We recommend submitting Ex*.mlx with the exported PDF (generated by the .mlx). Submit your assignment to the course moodle page in the form of a *.zip (**not RAR**) containing ***Ex3.mlx*** or ***Ex3.m*** - the main MATLAB file, other *.m files and images along with a report in the form of a PDF file (NOT .doc). **Both the PDF and ZIP file names should be the initials and ID of both of the team members ex. 'SC-1234567_RS-7654321.pdf' and 'SC-1234567_RS-7654321.zip', respectively.**

Academic integrity: the originality of the submitted exercises **will be checked.**

Document Instructions

- Only one of the team members should submit the file
- The report should be written in Hebrew or English.
- Each section should have the relevant title as is in this document.
- Every image should be accompanied with the relevant explanation.
- The displayed images should be large enough for us to see them.

- The document should be organized and readable.

Code Instructions

- Use MATLAB version 2014b or later. If you don't have one on your computer, you can work from the computer laboratories in building 33 using VPN.
- A **main** function should call all the section functions in the correct order and should be named ***Ex3.mlx*** or ***Ex3.m***.
- The first line of ***Ex3.mlx / Ex3.m*** should print the full names and IDs of all team members. Use MATLAB's *disp()* function.
- Write modular functions for the subsections and reuse those functions throughout your code whenever possible.
- Every **.m* file should start with a comment containing the full names and IDs of all team members.
- Use meaningful names for all functions and variables.
- Try to avoid overriding variables.
- Write comments for every line of code that is not completely self explanatory.
- For every image displayed give a meaningful title using MATLAB's title() function.
- Use subplots whenever possible.
- All paths to files should be relative paths. If you are using subfolders use MATLAB's *fullfile()* function to construct the path to the file. Do not hard code '/' or '\' in the paths.
- The code should run completely without errors. A project with errors **will not be checked!**