TEL AVIV UNIVERSITY

Image Processing Lab

2022-2023

Group #8

Experiment #4

Preliminary Report

Pair No. 43

ID1: 301613501

ID2: 208560086

1. Proofing of Fourier Transform Properties

- Proof for 2D Fourier Transform **Linearity** Property

$Fourier\ transform\ definition$:

$$F(\xi_1,\xi_2) \triangleq \mathcal{F}\{f(x,y)\} = \iint_{-\infty}^{\infty} f(x,y) * e^{-j2\pi(x\xi_1+y\xi_2)}\,dx\,dy$$

$Claim$:

$$\mathcal{F}\{a_1f_1(x,y)+a_2f_2(x,y)\} = a_1F_1(\xi_1,\xi_2)+a_2F_2(\xi_1,\xi_2)$$

$Proof$:

$Let\ us\ define,$

$$g(x,y) \triangleq a_1f_1(x,y)+a_2f_2(x,y)$$

$$\mathcal{F}\{g(x,y)\} = \iint_{-\infty}^{\infty} \left(a_1f_1(x,y)+a_2f_2(x,y)\right) * e^{-j2\pi(x\xi_1+y\xi_2)}\,dx\,dy$$

$$= \iint_{-\infty}^{\infty} a_1f_1(x,y) * e^{-j2\pi(x\xi_1+y\xi_2)}\,dx\,dy + \iint_{-\infty}^{\infty} a_2f_2(x,y) * e^{-j2\pi(x\xi_1+y\xi_2)}\,dx\,dy$$

$$= a_1 \iint_{-\infty}^{\infty} f_1(x,y) * e^{-j2\pi(x\xi_1+y\xi_2)}\,dx\,dy + a_2 \iint_{-\infty}^{\infty} f_2(x,y) * e^{-j2\pi(x\xi_1+y\xi_2)}\,dx\,dy$$

$$= a_1F_1(\xi_1,\xi_2)+a_2F_2(\xi_1,\xi_2)\ \blacksquare$$

- Proof for 2D Fourier Transform **Scaling** Property

$Claim$:

$$\mathcal{F}\{f(ax,by)\} = \frac{1}{|ab|}F\left(\frac{\xi_1}{a},\frac{\xi_2}{b}\right)$$

$Proof$:

$$\mathcal{F}\{f(ax,by)\} = \iint_{-\infty}^{\infty} f(ax,by) * e^{-j2\pi(x\xi_1+y\xi_2)}\,dx\,dy$$

$Denote,$

$$u \triangleq ax\ ;\ v \triangleq by; \qquad \frac{u}{a} = x;\ \frac{v}{b} = y \rightarrow \frac{du}{a} = dx;\ \frac{dv}{b} = dy$$

$Hence,$

$$\mathcal{F}\{f(ax,by)\} = \iint_{-\infty}^{\infty} f(u,v) * e^{-j2\pi\left(\frac{u}{a}\xi_1 + \frac{v}{b}\xi_2\right)} \frac{dx\,dy}{ab} = \frac{1}{|ab|} F\left(\frac{\xi_1}{a}, \frac{\xi_2}{b}\right)$$
$$= \frac{1}{|ab|} F\left(\frac{\xi_1}{a}, \frac{\xi_2}{b}\right) \blacksquare$$

The absolute value was added to ensure the result of the integral sign would not change

- Proof for 2D Fourier Transform **Rotation** Property

$Claim$:

$$\mathcal{F}\{f(\pm x, \pm y)\} = F(\pm\xi_1\,, \pm\xi_2\,)$$

$Proof$:

$Let\ us\ use\ the\ scaling\ property\ above\ for\ a = b = -1$

$Hence,$

$$\mathcal{F}\{f(-x,-y)\} = \frac{1}{|1|} F\left(\frac{\xi_1}{-1}, \frac{\xi_2}{-1}\right) = F(-\xi_1, -\xi_2) \blacksquare$$

According to the message in the lab forum, let us look also on the polar coordinate case for proofing the rotation:

Similarly, to the Cartesian coordinate, we will transform the frequency coordinate also.

$$F(\rho,\phi) \triangleq \mathcal{F}\{f(r,\theta)\} = \int_{\theta=0}^{2\pi} \int_{r=0}^{\infty} f_p(r,\theta) \cdot e^{-j2\pi r\rho\cos(\theta-\phi)} \cdot r \cdot dr \cdot d\theta$$

$Claim$:

$$\mathcal{F}\{f_p(r,\theta+\alpha)\} = F(\rho,\phi+\alpha)$$

$Proof$:

$Cartesian - Polar$:

$$x = r\cos(\theta+\alpha)\,; \qquad y = r\sin(\theta+\alpha); \qquad r = \sqrt{x^2+y^2}\,; \qquad \theta = \arctan\left(\frac{x}{y}\right)$$

$Frequency - Polar$:

$$\xi_1 = \rho\cos(\phi)+\alpha\,; \quad \xi_2 = \rho\sin(\phi+\alpha); \quad \rho = \sqrt{{\xi_1}^2 + {\xi_2}^2}\,; \quad \phi = \arctan\left(\frac{\xi_1}{y\xi_1}\right)$$

$$\mathcal{F}\{f(r,\theta+\alpha)\} = \int_{\theta=0}^{2\pi}\int_{r=0}^{\infty} f_p(r,\theta+\alpha)\cdot e^{-j2\pi r\rho\cos(\theta-(\phi+\alpha))}\cdot r\cdot dr\cdot d\theta$$

$$= F(\rho,\phi+\alpha)\ \blacksquare$$

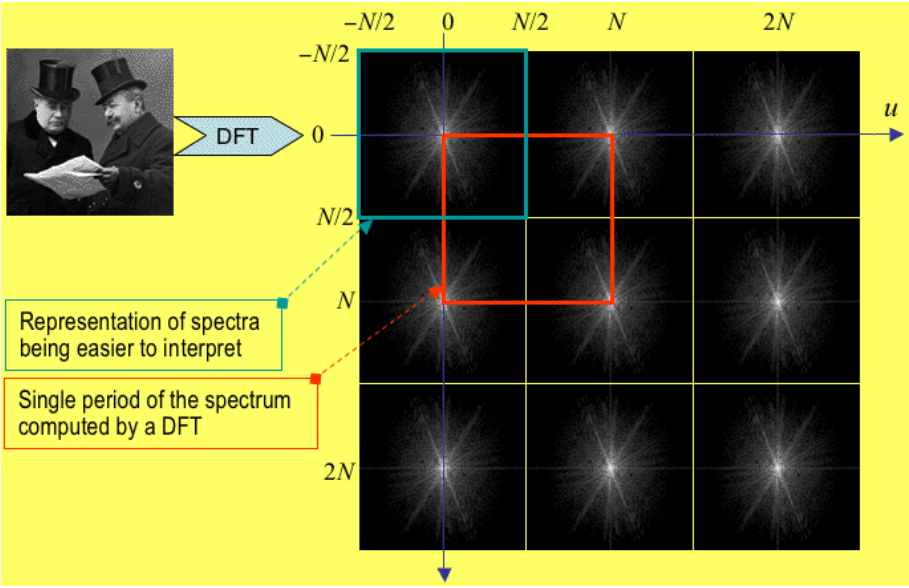
2. The periodicity of the Discrete Fourier Transform is given by

$$F[u+M,v+N] = F[u,v]$$

This property means that the image will repeat itself after N & M samples. So, in a similar way to the 1D periodicity of DFT, we know that in order to recover a signal and avoid aliasing we need to hold the Nyquist rule that the sampling frequency will be at least twice the original signal frequency. Here in image processing, we have to take it also into account to prevent destroying the sampled image by aliasing by the following Nyquist rule for 2D DFT:

$$\frac{1}{N} = 2\xi_1 \ and\ \frac{1}{M} = 2\xi_2$$

3. Complex conjugate symmetry means that $|F(u,v)| = |F(-u,-v)|$, so that there exist negative frequencies which are mirror images of the corresponding positive frequencies as regarding the amplitude spectrum. Due to periodicity and conjugate symmetry, the computed spectra have one peculiarity: for the computed values of $F(u,v)$, frequency increases up to $u = N/2$ and decreases thereafter; the half of the spectrum for which $u > N/2$ is a "double reflection" of the half with $u \leq N/2$, and the same applies to frequencies either side of $v = N/2$, as illustrated below (Link to Source):



4. As we usually do with wide ranges display, we are displaying graphs or spectrums in a log scale.
5. Explanations according to num

   - **numpy.fft.fft2**
     This function computes the *n*-dimensional discrete Fourier Transform over any axes in an *M*-dimensional array by means of the Fast Fourier Transform (FFT). By default, the transform is computed over the last two axes of the input array, i.e., a 2-dimensional FFT. Source
   - **numpy.fft.fftshift**
     Shift the zero-frequency component to the center of the spectrum. This function swaps half-spaces for all axes listed (defaults to all). Note that $y[0]$ is the Nyquist component only if $len(x)$ is even. Source

6. **cv2.getOptimalDFTSize** returns the minimum number N that is greater than or equal to vecsize so that the DFT of a vector of size N can be processed efficiently.
   The function returns a negative number if the size is too large (very close to INT_MAX). We would also note that while the function cannot be used directly to estimate the optimal vector size for DCT transform (since the current DCT implementation supports only even-size vectors), it can be easily processed as $getOptimalDFTSize((vecsize + 1)/2) * 2$. Source

7. 2D Rectangular Function
   a. Calculation of 2D Fourier Transform

$$\Pi(x,y) = \begin{cases} 1, & |x| \le \frac{1}{2}, |y| \le \frac{1}{2} \\ 0, & otherwise \end{cases}$$

$$\mathcal{F}\{\Pi(x,y)\} = \iint_{-\infty}^{\infty} \Pi(x,y) * e^{-j2\pi(x\xi_1 + y\xi_2)}\, dx\, dy$$

$$= \int_{-\frac{1}{2}}^{\frac{1}{2}} e^{-j2\pi(x\xi_1)} dx \int_{-\frac{1}{2}}^{\frac{1}{2}} e^{-j2\pi(y\xi_2)}\, dy$$

$$= \frac{1}{-j2\pi\xi_1}\left[e^{-j\pi\xi_1} - e^{j\pi\xi_1}\right]\frac{1}{-j2\pi\xi_2}\left[e^{-j\pi\xi_2} - e^{j\pi\xi_2}\right]$$

$$= \frac{1}{j2\pi\xi_1}\left[e^{j\pi\xi_1} + e^{-j\pi\xi_1}\right]\frac{1}{j2\pi\xi_2}\left[e^{j\pi\xi_2} - e^{-j\pi\xi_2}\right]$$

$$(*)\ Note\ that: \sin(x) = \frac{e^{jx} - e^{-jx}}{j2}$$

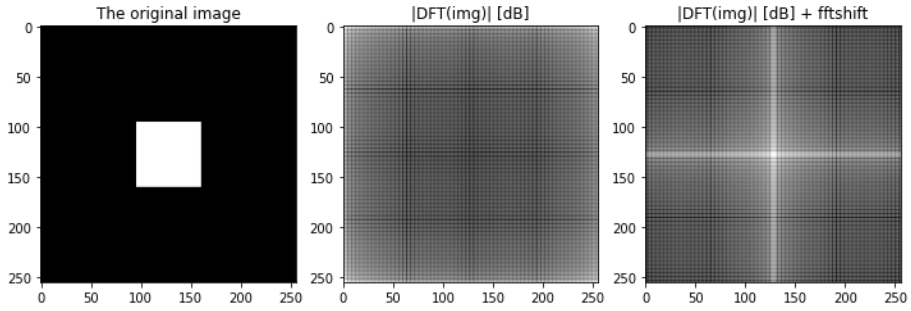
$$= \frac{\sin(\pi\xi_1)}{\pi\xi_1}\frac{\sin(\pi\xi_2)}{\pi\xi_2}$$

$$= sinc(\xi_1)sinc(\xi_1) = sinc(\xi_1, \xi_2)$$

b. A centered square window of dimensions 64x 64

```
W = 64 # window squared dimenstion
M = 256 # image squared dimenstion
img = np.zeros((M,M), int)
for i in range(M):
 for j in range(M):
  if ( (i >= ((M-W)/2)) & (i <= ((M+W)/2)) & (j >= ((M-W)/2)) & (j <= ((M+W)/2)) ):
   img[i,j] = 255
  else:
   img[i,j] = 0

ft = fft.fft2(img)
log_mag_ft = 20*np.log(np.abs(ft)+1)
ft_shift = fft.fftshift(ft)
log_mag_ft_shift = 20*np.log(np.abs(ft_shift)+1)
fig,ax = plt.subplots(nrows=1,ncols=3,figsize=(10,10))
ax = ax.ravel()
ax[0].imshow(img,cmap='gray')
ax[0].set_title('The original image'),ax[0].axis('on')
ax[1].imshow(log_mag_ft,cmap='gray')
ax[1].set_title('|DFT(img)| [dB]')
ax[2].imshow(log_mag_ft_shift,cmap='gray')
ax[2].set_title('|DFT(img)| [dB] + fftshift')
plt.tight_layout()
plt.show()
```



In the left image we see the original generated image. While in the middle image we can see the magnitude of the DFT in log scale before the shifting, so the we see the periodicity property of the transformation. In order to see the magnitude in the correct way we used the shifting function and then we can see the expected DFT of our

rectangular binary image → a cross-white that represent the edge of the rectangular by high frequencies representation.

8. 2D Circular Rectangular Function

Source: Goodman J.W. Introduction to Fourier Optics 2nd. McGraw-Hill 1996

$$circ(r,\theta;R)=\begin{cases}1, & r\le R\\ 0, & r>R\end{cases}$$

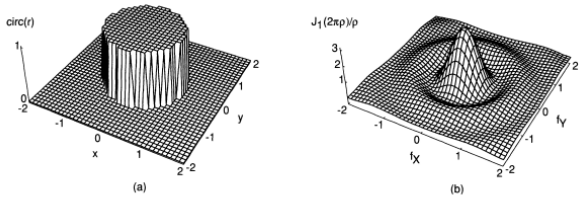


FIGURE 2.3
(a) The circle function and (b) its transform.

a. Calculation of 2D Fourier Transform

$Claim$:

$$\mathcal{F}_{2D}\{circ(r;1)\}=\frac{J_1(2\pi\rho)}{\rho}\triangleq\pi\,somb(\rho)$$

$Where,$

$J_1(\alpha)\ is\ the\ 1st\ order\ Bessel\ function\ defined\ as$

$$\alpha J_1(\alpha)=\int_0^{\alpha} xJ_0(x)\,dx$$

$$J_0(\alpha)=\frac{1}{2\pi}\int_0^{2\pi} e^{-j\alpha\cos(\theta-\phi)}\,d\theta$$

$Proof$:

$We\ can\ see\ that\ the\ circ\ function\ is\ not\ dependent\ in\ \theta$
$which\ means\ that\ it\ is\ a\ circular\ symetric\ function\ by\ definition$
$(according\ to\ lecture), there\ for\ the\ Fourier\ transform\ will\ be\ also$
$circular\ symetric\ function.$

$$F(\rho)\triangleq\mathcal{F}\{circ(r;1)\}=$$

$$=\int_{r=0}^{R} circ(r;1)\cdot e^{-j2\pi r\rho\cos(\theta-\phi)}\cdot r\cdot dr$$

$$= \int_{r=0}^{1} 1 \cdot e^{-j2\pi r\rho\cos(\theta-\phi)} \cdot r \cdot dr$$

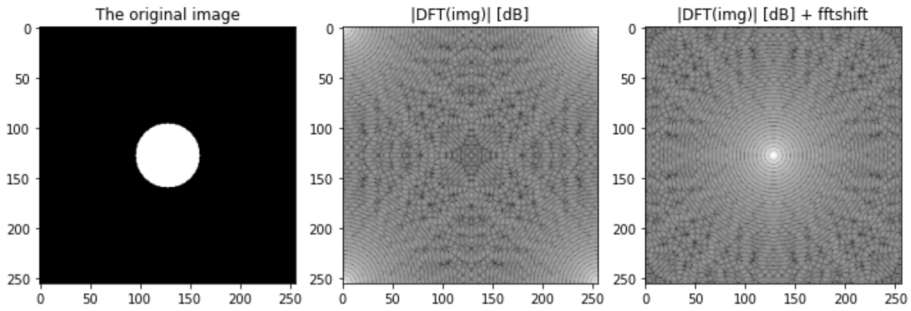
$$= \int_{r=0}^{1} 2\pi r \cdot J_0(2\pi r\rho) dr$$

$$Using\ change\ of\ variable\ \eta = 2\pi r\rho\ will\ give$$

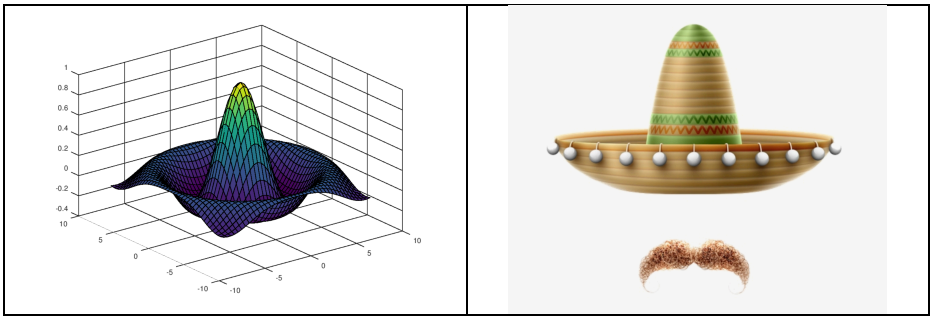
$$= \int_{\eta=0}^{2\pi\rho} \frac{2\pi\eta}{2\pi\rho} \cdot J_0(\eta) \frac{d\eta}{2\pi\rho} = \int_{\eta=0}^{2\pi\rho} \frac{\eta}{2\pi\rho^2} \cdot J_0(\eta) d\eta = \frac{J_1(2\pi\rho)}{\rho} \blacksquare$$

b. A centered circular window with $R = 32$

```
M = 256    # image squared dimenstion
R = 32
img = np.zeros((M,M),int)
for i in range(M):
 for j in range(M):
  r = np.sqrt((i - 128) ** 2 + (j - 128) ** 2)
  if ( (r <= R)):
   img[i,j] = 255
  else:
   img[i,j] = 0
```



c. This function is called “Sombrero” function because of its similarity to the Sombrero heat shape.



d. Yes, the major resemblance between the 2 DTs is that the high frequencies are concentrated in the center of the image of the FT. in addition we can see that the duplicated shapes have similar shapes according to the frequency gradient in the original images.

9. **scipy.ndimage and its filters**

The SciPy provides the ndimage (n-dimensional image) package, that contains the number of general image processing and analysis functions. It is dedicated to image processing.

Filter	**Explanation**
uniform_filter	Uniform filter (a.k.a. mean filter), which replaces the value of a pixel by the mean value of an area centered at the pixel.
median_filter	Median filter, which replaces the value of a pixel by the median value of an area centered at the pixel. This filter is better for edges recognitions.
gaussian_filter	Gaussian filter, which is similar to a mean filter but weighted in favor of pixels closer to the center.
Laplace	A Laplacian filter is an edge detector used to compute the second derivatives of an image, measuring the rate at which the first derivatives change.
Prewitt	The Prewitt operator detects image edges by convolution with two filter masks. One for horizontal and one for vertical direction.
Sobel	The Sobel filter is used for edge detection. It works by calculating the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction.
The most important parameter for using those filters is the kernel size of the filter.	

10. **skimage.filters.unsharp_mask**

The Unsharp Mask is designed to enhance the details in an image. It increases the image contrast along the edges of objects in an image.

The effect doesn't actually detect edges, but it can identify pixel values that differ from their neighboring pixels by a certain amount.

Unsharp masking is a linear image processing technique which sharpens the image.
The sharp details are identified as a difference between the original image and its blurred version.

These details are then scaled, and added back to the original image by the following equation (source):

$$enhanced\ image = original + amount * (original - blurred)$$

The blurring step could use any image filter method, e.g., median filter, but traditionally a gaussian filter is used in the skimage filters module. Such that, the radius parameter in the unsharp masking filter refers to the sigma parameter of the gaussian filter.

Wikipedia explains the process clearly by the following step by step illustration