Two-Dimensional Signal Processing and Image De-noising

Alejandro Ribeiro

March 14, 2021

Until now, we considered (one-dimensional) discrete signals of the form $x : [0, N-1] \to \mathbb{C}$ of duration N and with elements $x(n)$ for $n \in [0, N-1]$. We extend this definition to two dimensions by considering the set of ordered pairs $\mathbf{x} : [0, M-1] \times [0, N-1] \to \mathbb{C}$:

$$\mathbf{x} = \{x(m, n) : m \in [0, M-1], n \in [0, N-1]\}. \tag{1}$$

We can think of $\mathbf{x}$ as a set of values on the integer lattice in the two-dimensional plane, i.e., as elements of a discrete *spatial* domain. We associate these signals with the space of complex matrices $\mathbb{C}^{M \times N}$. The inner product for two signals $\mathbf{x}$ and $\mathbf{y}$, both of *size* $M \times N$, in two-dimensions is a natural extension of the one-dimensional case and is defined as

$$\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m, n) y^*(m, n). \tag{2}$$

As before, we define the energy of a two-dimensional signal $\mathbf{x}$ as $\langle \mathbf{x}, \mathbf{x} \rangle = \|\mathbf{x}\|^2$. We say two signals are orthogonal in two-dimensions if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ and $\mathbf{x} \neq \mathbf{y}$. If both vectors also have unit energy, i.e., $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$, they are said to be orthonormal. The discrete two-dimensional impulse $\delta(n, m)$ (Kronecker delta) is defined as

$$\delta(m, n) = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases} \tag{3}$$

The discrete complex exponential $\mathbf{e}_{kl,MN}(m, n)$ in two-dimensions of fre-

quencies k and l is the signal

$$\begin{aligned}\mathbf{e}_{kl,MN}(m,n) &= \frac{1}{\sqrt{MN}}\, e^{-j2\pi km/M} e^{-j2\pi ln/N} \\ &= \frac{1}{\sqrt{MN}}\, e^{-j2\pi(km/M+ln/N)}. \end{aligned} \tag{4}$$

It is a straightforward computation to check that $\mathbf{e}_{kl,MN}$ is orthonormal to $\mathbf{e}_{pq,MN}$ for $k \neq p$ and $l \neq q$.

The two-dimensional discrete Fourier transform (2-D DFT) of $\mathbf{x}$ is the signal $\mathbf{X} : \mathbb{Z}^2 \to \mathbb{C}$ whose the elements $\mathbf{X}(k,l)$ for all $k, l \in \mathbb{Z}$ are defined as

$$\mathbf{X}(k,l) := \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathbf{x}(m,n) e^{-j2\pi km/M} e^{-j2\pi ln/N} \tag{5}$$

The arguments k and l of the signal $\mathbf{X}(k,l)$ are called the vertical and horizontal frequency of the DFT and the value $\mathbf{X}(k,l)$ is referred to as the frequency component of $\mathbf{x}$ at (k,l). As in the one-dimensional case, when $\mathbf{X}$ is the DFT of $\mathbf{x}$ we write $\mathbf{X} = \mathcal{F}(\mathbf{x})$. Recall that for a complex exponential, the discrete frequency k is equivalent to the (real) frequency $f_k = (k/N) f_s$, where N is the total number of samples and f_s is the sampling frequency.

Notice that as in the one-dimensional case, we can interpret the 2-D DFT as a two-dimensional inner-product between $\mathbf{x}$ and the complex exponential $\mathbf{e}_{kl,MN}(m,n)$. Indeed, using the definition of the inner-product in (2) we can then write

$$\begin{aligned}\mathbf{X}(k,l) &:= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathbf{x}(m,n) e^{-j2\pi km/M} e^{-j2\pi ln/N} \\ &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathbf{x}(m,n) \left(\frac{1}{\sqrt{MN}} e^{-j2\pi km/M} e^{-j2\pi ln/N} \right) \\ &= \langle \mathbf{x}, \mathbf{e}_{kl,MN} \rangle. \end{aligned} \tag{6}$$

Therefore, we can view $\mathbf{X}(k,l)$ as a measure of how much the signal $\mathbf{x}$ resembles an oscillation of frequency k in the vertical direction and l in the horizontal direction.

Because the complex exponential is (M,N)-periodic, the 2-D DFT val-

ues $\mathbf{X}(k,l)$ and $\mathbf{X}(k+M,l+N)$ are equal, i.e.,

$$\begin{aligned}\mathbf{X}(k+M,l+N) &= \frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\mathbf{x}(m,n)e^{-j2\pi(k+M)m/M}e^{-j2\pi(l+N)n/N} \\ &= \frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\mathbf{x}(m,n)e^{-j2\pi km/M}e^{-j2\pi ln/N} \\ &= \mathbf{X}(k,l). \end{aligned} \tag{7}$$

The relationship in (7) means that the DFT is periodic in both directions with periods N and M respectively. While it is defined for all $k,l \in \mathbb{Z}^2$, only MN values are distinct. As in the one-dimensional case, we work with the canonical set of frequencies $k,l \in [0,M-1]\times[0,N-1]$ for computational purposes and the set $k,l \in [-M/2,M/2]\times[-N/2,N/2]$ for interpretation purposes. Notice that this latter set contains $(M+1)(N+1)$ frequencies instead of MN.

As before, we may shift the values of the 2-D DFT to convert from one canonical set to another using periodicity:

$$\mathbf{X}(-k,-l) = \mathbf{X}(M-k,N-l) \tag{8}$$

for all $k,l \in [-M/2,M/2]\times[-N/2,N/2]$. The operation in (8) is a "chop and shift," from which we may recover the DFT values for the canonical set $[-M/2,M/2]\times[-N/2,N/2]$ from the canonical set $[0,M-1]\times[0,N-1]$. For the purposes of this homework, when you are asked to report a DFT, you should report the DFT for the canonical set $[-M/2,M/2]\times[-N/2,N/2]$.

Finally, we define the 2-D inverse Discrete Fourier Transform (2-D iDFT) $\mathcal{F}^{-1}(\mathbf{X})$ as the two-dimensional signal $\tilde{\mathbf{x}}(m,n)$

$$\begin{aligned}\tilde{\mathbf{x}}(m,n) &:= \frac{1}{\sqrt{MN}}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\mathbf{X}(k,l)e^{j2\pi km/M}e^{j2\pi ln/N} \\ &= \frac{1}{\sqrt{MN}}\sum_{k=-M/2+1}^{M/2}\sum_{l=-N/2+1}^{N/2}\mathbf{X}(k,l)e^{j2\pi km/M}e^{j2\pi ln/N} \end{aligned} \tag{9}$$

The expression in (9) means that any arbitrary signal $\mathbf{x}$ in two dimensions may be represented as a sum of oscillations as in the one dimensional case. Moreover, conjugacy means we can represent the sum (9) with only half as many terms. This means that we can effectively represent 2-D signals (e.g., images) using only particular DFT coefficients under some conditions. This observation is the foundation of image de-noising and

compression methods which are the focus of this and next week's lab assignments. For the subsequent questions, **assume that** $M = N$ so that signals are of dimension N^2.

1 Two Dimensional Signal Processing

[**Note:** You may consider Python function `matplotlib.pyplot.imshow` to plot images.]

1.1 Inner products and orthogonality. Write a Python class that takes as input two-dimensional signals $\mathbf{x}$ and $\mathbf{y}$ and outputs their inner product. Each signal is defined by an $N \times N$ matrix of complex numbers.

1.2 Discrete Complex Exponentials. Write a Python class that takes as input the frequencies k, l and the signal duration N and returns three matrices of size $N \times N$, the first containing the complex values of $\mathbf{e}_{kl,NN}$ and the later two containing its real and imaginary parts separately.

1.3 Unit Energy 2-D Square Pulse. The two dimensional square pulse is defined as

$$\sqcap_L(m,n) = \begin{cases} \frac{1}{L^2} & \text{if } 0 \leq m,n < L \\ 0 & \text{if } m,n \geq L \end{cases} \tag{10}$$

Write a Python class that takes as input the size N, the size of the square pulse L and outputs a two-dimensional square pulse as a matrix of size $N \times N$ as well as the total number of samples N^2. Plot the two-dimensional square pulse for $N = 32$ and $L = 4$.

1.4 Two-Dimensional Gaussian Signals. An uncorrelated Gaussian pulse centered at μ is defined as

$$\mathbf{G}_\sigma(m,n) = e^{-[(m-\mu)^2+(n-\mu)^2]/(2\sigma^2)}. \tag{11}$$

Write a Python class that takes as input N, μ and σ, and outputs a two-dimensional Gaussian pulse. Plot the two dimensional Gaussian for $N = 9$, $\mu = 2$ and $\sigma = 1$, for $N = 65$, $\mu = 21$ and $\sigma = 10$ and for $N = 255$, $\mu = 128$ and $\sigma = 42$. Note what these signals look like when projected onto a single dimension.

1.5 DFT in two dimensions. Modify your Python class for the one dimensional DFT from lab 2 so that it now computes the 2-D DFT. Recall that this computation can be expressed in terms of an inner product (using the function written for question 1.1) between the signal and a two-dimensional discrete complex exponential (see 1.2). Compute the DFT of a 2-D Gaussian pulse with $\mu = 0$ and $\sigma = 2$. Plot your results on the two-dimensional plane.

1.6 iDFT in Two Dimensions. Write a Python class which takes as input an $N \times N$-dimensional signal and computes its 2-D iDFT as in (9). Exploit conjugacy in your computation so that you only need to take in $N^2/2$ DFT coefficients (recall that $X(-k,-l) = X^*(k,l)$ and that $X(k,-l) = X^*(-k,l)$). Compute the iDFT associated with the DFT of the Gaussian pulse you computed in the previous question. Plot this signal and compare it with the original Gaussian pulse.

2 Image Filtering and de-noising

We can de-noise corrupted images by using spatial information about the signal. In the case of images, we know pixels that are close do not change value too much. We can therefore get rid of noise by averaging the value of nearby pixels. Recall that averaging is a form of low-pass filtering. In this sense, we will de-noise 2-D signals the same way we de-noised 1-D signals: by filtering. As before, filtering is represented as a convolution with the filter impulse response. In the two-dimensional case, the convolution of two signals $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$\begin{aligned}[\mathbf{x} * \mathbf{y}](m,n) :&= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k,l)y(m-k,n-l) \\ &= \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} x(k,l)y(m-k,n-l) \end{aligned} \tag{12}$$

In image processing, it is common to use Gaussian filters for de-noising. In other words, we convolve the image with the Gaussian pulse $\mathbf{G}_\sigma$ from (11) to obtain

$$\tilde{\mathbf{x}}_{\text{de-noised}} = \mathbf{G}_\sigma * \mathbf{x}. \tag{13}$$

2.1 Spatial de-noising. Implement the Gaussian filtering (13) without using the 2-D DFT by directly convolving the images provided with a Gaussian pulse. Use $\mu = (N-1)/2$ and $\sigma = (N-1)/6$ in the function from 1.4 so your pulse is centered and the filter has size $N \times N$. Try your implementation for $N = 7, 13, 25$ on sample images A and B. Do you observe significant de-noising performance differences when varying σ? *Hint:* look up the Python functions `matplotlib.image.imread` and `scipy.signal.convolve2d`.