Lecture 12

DFT in Multiple Dimensions

12.1 Introduction

Extension of the DFT to the N-dimensional case results in representing our N-D signals as a linear combination of N-D complex exponentials

$$e^{i2\pi\frac{m_0k_0+\cdots+m_{N-1}k_{N-1}}{M}} = e^{i2\pi\frac{m_0k_0}{M}}\cdots e^{i2\pi\frac{m_{N-1}k_{N-1}}{M}}$$

.

12.2 Definition of the ND DFT

The DFT of an array in N dimensions, each with M elements $f[m_0, m_1, ..., m_{N-1}]$ is defined as:

$$\hat{f}[k_0,\ldots,k_{N-1}] = \sum_{m_0=0}^{M-1}\cdots\sum_{m_{N-1}=0}^{M-1} f[m_0, m_1, ..., m_{N-1}]e^{-i2\pi\frac{m_0k_0+\cdots+m_{N-1}k_{N-1}}{M}} \tag{12.1}$$

with N-D inverse DFT (iDFT) given by:

$$f[m_0,\ldots,m_{N-1}] = \frac{1}{M^N}\sum_{k_0=0}^{M-1}\cdots\sum_{k_{N-1}=0}^{M-1} \hat{f}[k_0, k_1, ..., k_{N-1}]e^{i2\pi\frac{m_0k_0+\cdots+m_{N-1}k_{N-1}}{M}} \tag{12.2}$$

Note that the extension to arrays with different numbers of elements along different dimensions is straightforward. Also, the specific case of two-dimensional DFT for an $M \times M$ array is given by:

$$\hat{f}[k_0, k_1] = \sum_{m_0=0}^{M-1}\sum_{m_1=0}^{M-1} f[m_0, m_1]e^{-i2\pi\frac{m_0k_0+m_1k_1}{M}} \tag{12.3}$$

with 2D iDFT:

$$f[m_0, m_1] = \frac{1}{M^2}\sum_{k_0=0}^{M-1}\sum_{k_1=0}^{M-1} \hat{f}[k_0, k_1]e^{i2\pi\frac{m_0k_0+m_1k_1}{M}} \tag{12.4}$$

12.3 Computation of the N-D DFT

Similar to the continuous Fourier transform, the DFT can be calculated by operating sequentially along each of the dimensions. An intuitive way to understand the calculation of the DFT in multiple dimensions is to formulate it as a DFT along the first dimension (keeping all other dimensions constant), followed by a DFT along the second dimension, etc. This is illustrated graphically in figure 12.1.

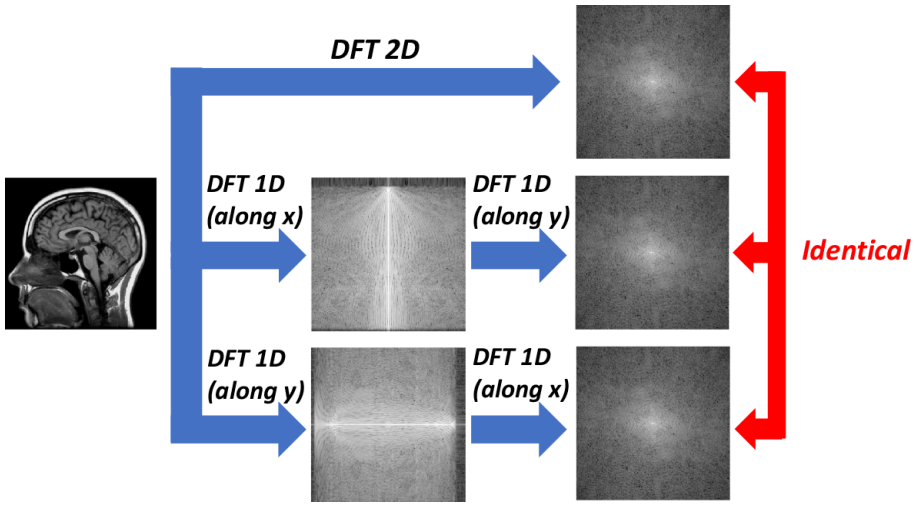

Figure 12.1: The 2D DFT can be equivalently decomposed as a 1D DFT along one dimension, followed by a 1D DFT along the other dimension. This equivalence extends to N-D DFTs, which can be decomposed into a concatenation of 1D DFTs along each of the N dimensions.

Question: What is the computational complexity (number of operations such as multiplications) required to calculate a 2D DFT naïvely (by directly implementing the definition shown above)? What is the number of operations required to calculate a 2D DFT using FFTs?

Question: Now assume that you have a 2D array with dimensions 10000×10000, and your computer requires $t_{op} = 10^{-9}$ seconds to calculate a single multiplication. What is the approximate time required to calculate your 2D DFT naïvely vs using FFTs? Please express each required time in the most appropriate units (seconds, hours, years, etc).

12.4 Example of N-D DFT

Question: Consider a small image of size 4×4, where the image is constant, ie: $f[m_0, m_1] = 1$, for $m_0 = 0, 1, 2, 3$ and $m_1 = 0, 1, 2, 3$. What is the 2D DFT of f? Please go through the calculations by hand, although you may exploit the various redundancies in the process (eg: first calculate the DFT along one dimension, and then along the other).