# Signal and Information Processing

Alejandro Ribeiro

April 5, 2016

# Contents

# Chapter 1

# Image Processing

## 1.1 Signal Representation

So far, we have been working with 1D signals in time, $x(n)$, indexed by a single time index $n$. These signals have been represented in 2 ways, as sums of shifted deltas, and as sums of oscillations.
A sum of shifted deltas

$$x(n) = \sum_{k=1}^{N} x(k)\delta(k-n) \tag{1.1}$$

is the representation of the signal in the time domain, where $x(k)\delta(k-n)$ is a single value of the signal at time $k$. Processing the values in the time signal one time step at a time is a very intuitive way of representing the signal, but a frequency domain representation is often more useful.
A sum of oscillations

$$x(n) = \sum_{k=1}^{N} X(k)e^{j2\pi kn/N} \tag{1.2}$$

is the representation in the frequency domain. With this representation, we can filter noise and classify sounds, operations we cannot do in the time domain.

Time and DFT representations are not the only ways we can represent signals, and 1D signals in time are not the only signals we can work with. For example

- 2D images with multidimensional DFTs or Discrete Cosine Transforms (DCT)
- Stochastic processes with Principal Component Analysis(PCA)
- Graph signal processing

Whenever we change how we represent a signal, we say that we change the basis. In linear algebra, this is changing the basis vectors used to span all possible signals in a vector space. With the right basis, we can perform operations and visualizations not previously possible.

(a) Black and white image representation

(b) Color image representation

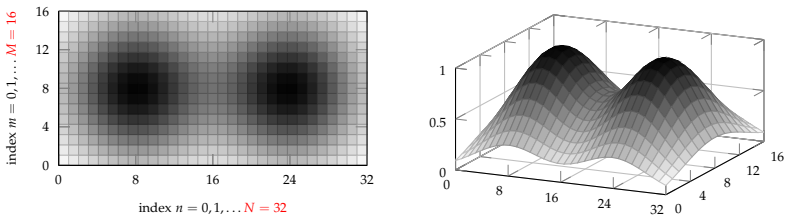


Figure 1.2: Two dimensional $M \times N$ signal

## 1.2 Images as Signals

Although in 2D, images are still signals and many of our 1D signal processing techniques can be applied with some modifications. Images are represented as a grid of pixels, each containing values representing color and brightness.

For a black and white image like in 1.1a, each pixel contains a single value, representing the luminescence of the point. For a color image like in 1.1b, more information about the image has to be represented. Each pixel contains multiple channels for different colors. For example, in a RGB (red, green, blue) image, each pixel is represented by 3 channels, while for a CMYK (cyan, magenta, yellow, black) image, each pixel is represented by 4 channels. Each of these channels can also form a 2D image on their own (but with a single color). Therefore, the more complex colored image is really just made up of multiple images, each representing a color channel. If we disregard the color and just focus on the luminescence, we see that these channels are no different from black and white images.

Although images are in 2D, they can be represented as signals with two sets of indices instead of one, as seen in figure 1.2

This 2D discrete signal is indexed by $(m, n)$

$$m = 0, 1, \ldots, M-1 = [0, M-1] \tag{1.3}$$
$$n = 0, 1, \ldots, N-1 = [0, N-1] \tag{1.4}$$

Formally, the signal is defined as a function that maps pairs of indices to real values.

$$x : [0, M-1] \times [0 : N-1] \to \mathbb{R}$$

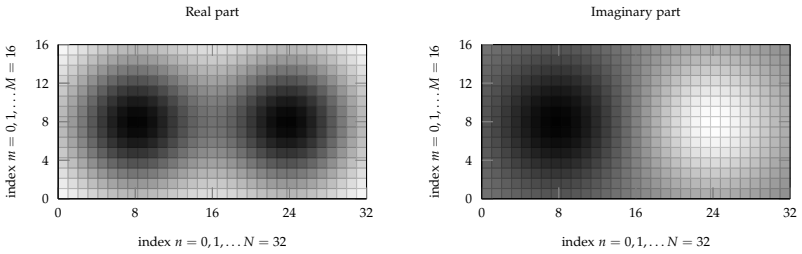


Figure 1.3: Complex 2D signals

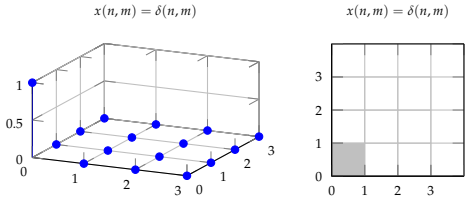


Figure 1.4: 2D Delta function

At indices $m$ and $n$, the signal $x$ has the value $x(m, n)$.
We can also have complex signals as well, with $x$ mapping to complex values.

$$x : [0, M-1] \times [0 : N-1] \to \mathbb{C}$$

Having complex 2D signals allows us to define and work with two dimensional DFTs, which involve 2D complex exponentials, seen in figure 1.3
We will now define 2D analogues of common 1D signals we have worked with before

### 1.2.1 Deltas in Two Dimensions

A 2D delta function $\delta(m, n)$ is simply a spike at $(m, n) = 0$, similar to how a 1D delta function $\delta n$ is a spike at $n = 0$.

$$\delta(m, n) = \begin{cases} 1 & \text{if } m = n = 0 \\ 0 & \text{else} \end{cases} \tag{1.5}$$

Similarly, a shifted delta $\delta(m - m_0, n - n_0)$ has a spike at $(m, n) = (m_0, n_0)$.

$$\delta(m - m_0, n - n_0) = \begin{cases} 1 & \text{if } (m, n) = (m_0, n_0) \\ 0 & \text{else} \end{cases} \tag{1.6}$$

### 1.2.2 Rectangular Pulses

A rectangular pulse with $N$ rows and $N$ columns, $\sqcap_{M_0 N_0}$ is defined as

$$\sqcap_{M_0 N_0}(m, n) = \begin{cases} 1 & \text{if } m < M_0, n < N_0 \\ 0 & \text{else} \end{cases} \tag{1.7}$$

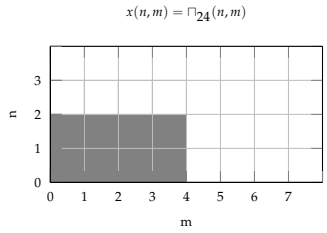


Figure 1.5: 2D Rectangular Pulse

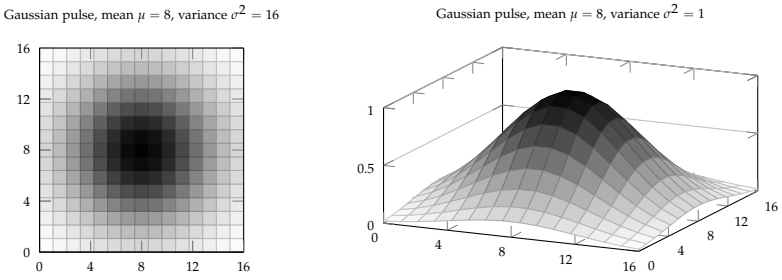


Figure 1.6: 2D Gaussian Pulse

We can also define square pulses when $M_0 = N_0$ as $\sqcap_{N_0 N_0} = \sqcap_{N_0}$, as well as shifted pulses $\sqcap_{M_0 N_0}(m - m_0, n - n_0)$, which moves the bottom-left corner of the pulse to $(m_0, n_0)$. However, to ensure that the entire pulse remains inside the $M \times N$ signal, we must have $m_0 < M - M_0$ and $n_0 < N - N_0$.

### 1.2.3 Gaussian Pulses

A symmetric 2D gaussian pulse with mean $\mu$ and variance $\sigma^2$ is defined as

$$g_{\mu\sigma}(m,n) = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{m-\mu}{2\sigma^2} - \frac{n-\mu}{2\sigma^2}\right] \tag{1.8}$$

This gives us a bell-shaped pulse centered at $(\mu, \mu)$, with 2D symmetry. The variance, $\sigma^2$, controls how fast the pulse decays as we move away from the center.

We can also create skewed gaussian pulses by adding a covariance matrix. It is easier to represent the equation for a generic gaussian pulse using matrix algebra. Therefore, we define the following matrices

- Coordinate Vector, $\mathbf{n} = [m, n]^T$, represents the location $(m, n)$.
- Center vector, $\mathbf{n} = [\mu_1, \mu_2]^T$, represents the center coordinates $(\mu_1, \mu_2)$
- Covariance matrix, $\mathbf{C} = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12} \\ \sigma_{12} & \sigma_{22}^2 \end{pmatrix}$

The convariance matrix controls how the gaussian pulse is stretched horizontally or vertically, determined by the value of the variances along the diagonal.

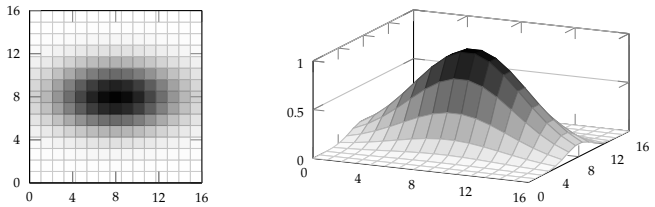


Figure 1.7: Gaussian pulse skewed in $m$ direction

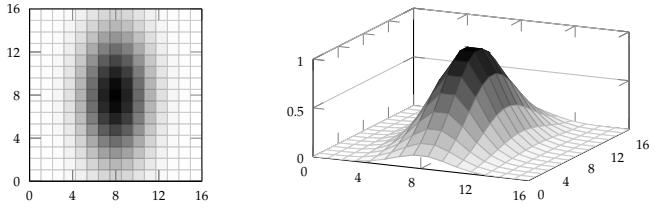


Figure 1.8: Gaussian pulse skewed in $m$ direction

The equation for a 2D gaussian pulse can now be expressed with matrix algebra as

$$g_{\mu\sigma}(n,m) = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{1}{2}(\mathbf{n}-\bar{})^{\mathbf{T}}\mathbf{C}^{-\mathbf{1}}(\mathbf{n}-\bar{})\right] \tag{1.9}$$

With $\mathbf{C} = \begin{pmatrix} 16 & 0 \\ 0 & 4 \end{pmatrix}$, we have a gaussian pulse in figure 1.7 skewed in the $m$ direction. Meanwhile, with $\mathbf{C} = \begin{pmatrix} 4 & 0 \\ 0 & 16 \end{pmatrix}$, we have a gaussian pulse in figure 1.8 skewed in the $n$ direction. Notice that $\sigma_{11}^2$ in the covarience matrix determines the spread along the $m$ axis, while $\sigma_{22}^2$ determines the spread along the $n$ axis. Therefore, $\sigma_{11}^2 > \sigma_{22}^2$ results in greater spread along $m$ than $n$ and a skew in the $m$ direction.

### 1.2.4 Inner Product

The inner product, which we have been using on 1D signals, can also be modified to work on 2D signals. We define the inner product in 2D as

$$\langle x, y\rangle = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x(m,n)y^*(m,n) \tag{1.10}$$

The inner product has the same properties in 2D as it does in 1D. Linearity, $\langle x, y+z\rangle = \langle x, y\rangle + \langle x, z\rangle$, and reversal of order, $\langle y, x\rangle = \langle x, y\rangle^*$, both behave the same way.

The 2D inner product can also be interpreted in a similar way to the 1D inner product. A positive result represents positive correlation, and the two signals tend to be similar. A negative result indicates that the two signals tend to be opposites of each other, and a 0 result indicates that the two signals are orthogonal.

For example, the inner product of 2 rectangular pulses, as seen in figure 1.9

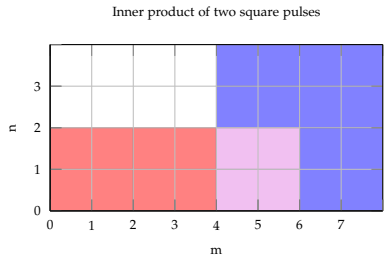


Figure 1.9: Inner product of rectangular pulses

is area where both pulses are not null, and therefore the sum of the overlap area of the 2 rectangles.

### 1.2.5 Norm and Energy

The norm of a 2D signal is defined as

$$||x|| = \left[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|x(m,n)|^2\right]^{1/2} \tag{1.11}$$

and the energy of a 2D signal is just the square of the norm

$$||x||^2 = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|x(m,n)|^2 = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|x_R(m,n)|^2 + \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|x_I(m,n)|^2 \tag{1.12}$$

Now that we have the definition of an inner product in 2D, we can also write the energy as an inner product, $||x||^2 = \langle x, x\rangle$.

As an example, assume we have a rectangular pulse $\sqcap_{M_0N_0}$ with $N_0$ rows and $M_0$ columns, as seen in figure 1.5

Using the definition for energy

$$||\sqcap_{M_0N_0}||^2 = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|\sqcap_{M_0N_0}|^2 \tag{1.13}$$

$$= \sum_{m=0}^{M_0-1}\sum_{n=0}^{N_0-1}1^2 \tag{1.14}$$

$$= M_0N_0 \tag{1.15}$$

We see that the energy of a rectangular pulse is equal to the number of pixels $(M_0N_0)$ in the pulse. With this information, we also see that we need to multiply a square pulse by $1/\sqrt{M_0N_0}$ in order to normalize its energy.

## 1.3 2D DFT

### 1.3.1 Definition

A 2D signal $x$ is defined as having $N$ rows and $M$ columns. For easier computation, we restrict our attention to the case where $N = M$. We call signals where $N = M$ **square** signals. The definition of the 2D DFT where $N = M$ is:

$$\frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)e^{-j2\pi(km+ln)/N} \tag{1.16}$$

Our notation is the same as in the 1D case. We write the 2D DFT of $x$ as $X = \mathcal{F}(x)$. We refer to $k$ as the **horizontal frequency** and $l$ as the **vertical frequency**.

### 1.3.2 Relating the 2D DFT to the 1D DFT

The 2D DFT may look complex in formal notation - but it is really just an extension of the vanilla 1D DFT. To see this in action we separate the summation terms of the 2D DFT equation and regroup the factors to get:

$$X(k,l) := \frac{1}{\sqrt{N}}\sum_{m=0}^{N-1}\left[\frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x(m,n)e^{-j2\pi ln/N}\right]e^{-j2\pi km/N} \tag{1.17}$$

Say we fix the outer value of m to a single value. Then the term in the parentheses is just the 1D DFT of $x(m_{fixed}, \cdot)$. The 2D DFT can be seen as the 1D DFT of a set of 1D DFTs. We say that this is the column-wise DFT of the row-wise DFTs - or the row-wise DFT of the column-wise DFTs (summation is order invariant so both are correct).

### 1.3.3 2D Discrete Complex Exponentials

The DFT is just a method to determine how complex exponentials of different frequencies can be used to compose a signal. This of course extends to the 2D case, except for now the complex exponentials are two dimensional. A 2D complex exponential has two frequency parameters: **horizontal frequency** $k$ and **vertical frequency** $l$. The definition is:

$$e_{klN}(m,n) = \frac{1}{N}e^{j2\pi(km+ln)/N} = \frac{1}{\sqrt{N}}e^{j2\pi(km/N)}\frac{1}{\sqrt{N}}e^{j2\pi(ln/N)} = e_{kN}(m)e_{lN}(n) \tag{1.18}$$

The 2D complex exponential is shown to be the product of two 1D complex exponentials with frequencies $k$ and $l$ respectively.

### 1.3.4 2D DFT as Inner Product

As in the 1D case, the 2D DFT can be written in inner product form.

We first rewrite the 2D DFT equation using the definition of the 2D complex exponential.

$$X(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)e_{(-k)(-l)N}(m,n) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)e^*_{klN}(m,n) \tag{1.19}$$

We can use the definition of the inner product to say that:

$$X(k,l) = \langle x, e_{klN} \rangle \tag{1.20}$$

Each element of the DFT $X(k,l)$ represents exactly how much of $x$ is an oscillation of horizontal frequency $k$ and vertical frequency $l$.

## 1.4 2D DFT of Images

Images are defined as two-dimensional grids (matrices) of pixels where each pixel takes on a color. It is immediately clear that images are the perfect discrete 2D signal to analyze with the 2D DFT. When we take the 2D DFT of an image, we always split the image into patches and take the 2D DFT of each patch. Why do we do this? The first reason is computational - it is much more efficient to split the image up and take the DFT of each patch than to take the DFT of the whole image. The more important reason is we actually get more information by looking at small patches. For instance, we can often determine whether a patch is part of the background or main content of the image by analyzing it's 2D DFT.

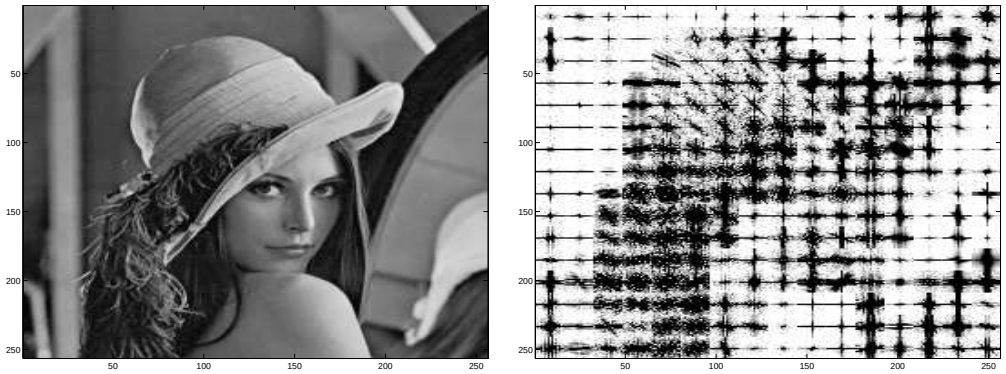


Figure 1.10: Lena and 2D DFT of patches

We can see in the 2D DFT of the image above that the background of the image has DFT patches with low variability. In visual terms, low variability means the DFT coefficients make one clean shape - e.g. a single vertical or diagonal line. Likewise, the DFT

coefficients of the image subject have high variability. In other words, the DFT patches representing the subject are messy and have no clear shape. This is a very useful insight and demonstrates why we separate an image into patches before taking the 2D DFT; we can tell visually from the 2D DFT what is likely content and what is likely background. This insight would not be possible if we took the DFT of the complete image.

## 1.5 Properties

### 1.5.1 Periodicity of Complex Exponentials

For a 2D square signal of dimension $N \times N$, there are $N^2$ distinct 2D complex exponentials. This extends from the 1D case. Horizontal frequencies $k$ and $k+N$ are equivalent. Likewise, vertical frequencies $l$ and $l+N$ are equivalent. Furthermore, 2D complex exponential are conjugate symmetric.

$$e_{(-k)(-l)N} \equiv e*_{klN} \tag{1.21}$$

### 1.5.2 Periodicity of the 2D DFT

In the previous subsection we asserted that 2D complex exponentials are periodic. We use this periodicity to establish the periodicity of the 2D DFT. The 2D DFT is shown below to be $N$ periodic in both the horizontal and vertical directions. Thus, if we have $N \times N$ adjacent frequencies, we have a completely defined frequency response. The two canonical sets we will use most often are $[0, N-1] \times [0, N-1]$ and $[-N/2, N/2] \times [-N/2, N/2]$.

$$X(k+N,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)e^*_{(k+N)lN}(m,n) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)e^*_{klN}(m,n) = X(k,l) \tag{1.22}$$

### 1.5.3 Orthogonality of complex exponentials

We assert that complex exponentials with non-equivalent frequencies are **orthogonal**. We demonstrate orthogonality using the inner product and showing it is zero in all cases accept when the frequencies of the two complex exponentials are the same.

$$\langle e_{klN}, e_{pqN} \rangle = \frac{1}{N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} e^{j2\pi km+ln)/N}\left(e^{j2\pi(\tilde{k}m+\tilde{l}n)/N)}\right)^* \tag{1.23}$$

$$= \frac{1}{N}\sum_{m=0}^{N-1} e^{j2\pi km/N}\left(e^{j2\pi\tilde{k}m/N}\right)^* \frac{1}{N}\sum_{n=0}^{N-1} e^{j2\pi ln/N}\left(e^{j2\pi\tilde{l}n/N}\right)^* \tag{1.24}$$

The first grouping is a delta of the horizontal frequencies. The second grouping is a delta of horizontal groupings. Therefore we have expression below - the delta functions

give that complex exponentials with non-equivalent frequencies are orthogonal.

$$\langle e_{klN}, e_{\tilde{k}\tilde{l}N} \rangle = \delta(k - \tilde{k})\delta(l - \tilde{l}) \tag{1.25}$$

## 1.6 2D iDFT

### 1.6.1 Definition

Given a Fourier transform $X$, the inverse (i)DFT is $x = \mathcal{F}^{-1}(X)$. We demonstrated before that the 2D DFT is $N$ periodic. This property applies to the 2D iDFT as well. Any summation over $N \times N$ adjacent frequencies will produce the 2D iDFT.

$$x(m,n) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} X(k,l) e^{j2\pi(km+ln)/N} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} X(k,l) e_{klN}(k,l) \tag{1.26}$$

We can also use the inner product form we derived for the 2D DFT. In this case we take the inner product of the DFT with the complex exponential of corresponding negative frequencies so the mathematical expression evaluates.

$$x(m,n) = \langle X, e_{(-k)(-l)N} \rangle \tag{1.27}$$

### 1.6.2 iDFT is inverse of DFT

For the complete proof refer to the slides. The proof that the iDFT is the inverse of the DFT is verbose and full of algebra and calculus. Really, this proves something that should be incredibly intuitive: the iDFT is the inverse of the DFT. If take the inverse DFT of the DFT of a 2D signal, we have recovered the original signal.

$$x = \mathcal{F}^{-1}(\mathcal{F}(x)) \tag{1.28}$$

### 1.6.3 Image Reconstruction

As described earlier, we split an image into patches before taking the 2D DFT of each patch. To recover the exact original image, we take the 2D iDFT of the 2D DFT of each patch and stitch them together like a quilt.

Let's say we take the same approach to compression we used in the 1D case. Recall that in 1D we took the DFT of a signal and kept the $k$ largest coefficients. By performing the iDFT using only the $k$ largest coefficients, we got an output signal with almost all of the energy of the original signal. Put colloquially, we were able to compress the original signal, yielding a new nearly identical signal while storing a fraction of the information. You can see how this is useful to all telecommunications.

Compression becomes more complicated in the case of images because the output is decoded **visually**. Even if the compressed signal has nearly the same energy and all of the

information as the original signal, if someone can visually register a difference none of that matters. The major drawback to using the DFT for image compression is that the DFT produces **BORDER EFFECTS**. If we view the image signal as a quilt, we want to sew the compressed patches back together without seeing the seams. This is not possible with the DFT unless we use all of the coefficients - but then we haven't compressed. The Discrete Cosine Transform described in a later section addresses this issue.

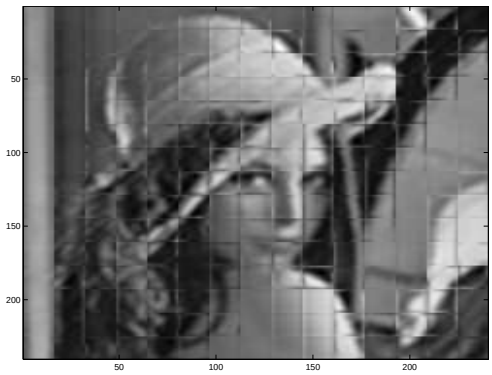


Figure 1.11: DFT Compressed Lena with Border Effects

## 1.7 Properties of the 2D DFT

All properties of 1D DFTs have corresponding versions for 2D DFTs. This includes linearity, conjugate symmetry, and the modulation shift duality. The two most important properties, energy conservation and the convolution/multiplication duality, will be discussed in further detail below.

### 1.7.1 Energy Conservation - Parseval's Theorem

The energies of a signal and its 2D DFT $X = \mathcal{F}(x)$ are the same. Refer to the lecture slides for a complete proof.

$$\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}|x(m,n)|^2 = \|x\|^2 = \|X\|^2 = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}|X(k,l)|^2 \tag{1.29}$$

This is exactly the same as in 1D case. Because the 2D DFT is periodic, any set of adjacent frequencies will satisfy this theorem.

$$\|X\|^2 = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|X(k,l)|^2 = \sum_{k=-M/2+1}^{M/2}\sum_{l=-N/2+1}^{N/2}|X(k,l)|^2 \tag{1.30}$$

### 1.7.2 Image Reconstruction Energy

When we reconstruct an image using all of its DFT coefficients, the recovered image has one hundred percent of the original image energy. When we reconstruct an image using less coefficients (i.e. $k$ coefficients), the image will contain less energy than the original image. This energy loss is known as the **reconstruction error** and is an excellent metric of how similar the compressed image is to the original image.

An example is shown below. Separate the original image into $16 \times 16$ patches. Compute the 2D DFT of each patch. Now compute the iDFT of each patch using only 4 coefficients. The new image contains sixty eight percent of the original image. The reconstruction error is 32%.

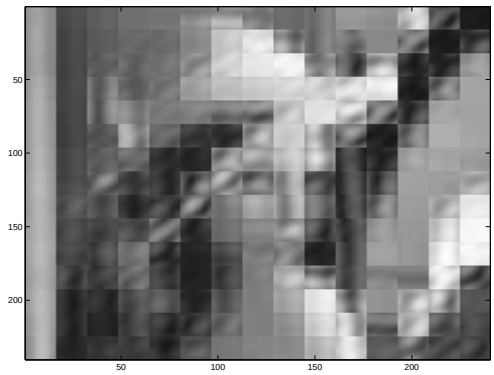


Figure 1.12: DFT Compressed Lena with 32% Reconstruction Error

### 1.7.3 2D Filtering

As in the 1D case, we can apply filters to 2D signals to control output behavior. The fundamental operation of filters is the **convolution**. Why the convolution is necessary to design filters won't be discussed in this course - take it on faith that convolutions and filters go hand in hand.

Say we have have an 2D input signal of dimension $N \times N$ and 2D filter of dimension $M \times M$. When we say the filter has dimension $M \times M$, we mean that the filter has value zero outside of the bounds: $[0, M-1] \times [0, M-1]$. The formula for the 2D convolution of the signal and filter is:

$$y(m,n) = \sum_{p=0}^{N} \sum_{q=0}^{N} x(p,q)h(m-p, n-q) \tag{1.31}$$

The result of hitting the $N \times N$ signal with the $M \times M$ filter is an $N + M \times N + M$ output.

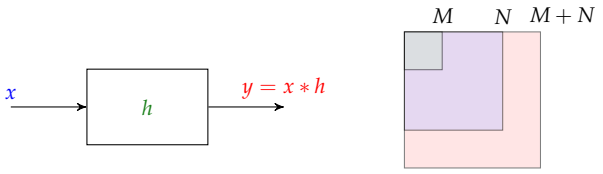


Figure 1.13: Result of Input Signal 'hit' by Filter

At this point we should be satisfied, but it's awkward to have a system where the inputs have dimensions $N \times N$ and $M \times M$ with output $N + M \times N + M$. For the sake of computation, it would be much easier to deal with a system where the inputs and output have the same dimension. All we have to do to achieve this is **pad** our input signals to both have dimension $N + M \times N + M$. Now the inputs and output of the system will all have dimension $N + M \times N + M$.

To pad a signal, create a new signal that retains the values of the original signal in the original bounds and has zeros elsewhere. In the figure below, the smaller squares represent the original bounds while the larger squares are the bounds of the new padded signals. The area between the small square and large square - i.e. the padding - contains zeros. Notice that both padded signals now have dimension $N + M \times N + M$.

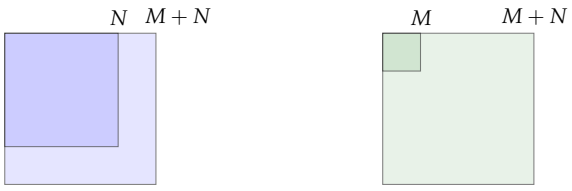


Figure 1.14: Padded signals

Now we get to the convolution theorem. **Convolution in space is multiplication in frequency. Multiplication in space is convolution in frequency**. To apply a filter in space we take the convolution. It is equivalent, and far easier computationally, to apply the filter in frequency by multiplying the DFT of the input signal with the DFT of the filter.

$$y = \bar{x} * \bar{h} \qquad \Longleftrightarrow \qquad Y = \bar{X}\bar{H} \tag{1.32}$$

As always you should be asking what the point of all this is. The answer is very simple: **we design our filters in the frequency domain and implement them in space**. In plain English, it's much easier to design filters by looking at Fourier transforms than it is looking at input signals. Once we have the filter designed, implementing it in space is trivial. We want to make our lives as easy as possible by exploiting the frequency domain for the harder task of designing the filter. An excellent graphic demonstrating this is shown below:

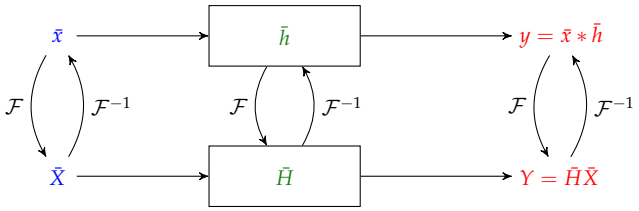


Figure 1.15: Design in Frequency, Implement in Space

The only trick is that we padded our inputs $x$ and $h$. In the notation, $\bar{x}$ and $\bar{h}$ are the padded versions of $x$ and $h$. Because $x * h$ and $\bar{x} * \bar{h}$ are equivalent, we can say $\bar{Y} \approx HX$.

There are a lot of really cool application to image filtering including every effect on Instagram. Things like noise filtering, image sharpening, edge detection, and image smoothing are all implemented with filters. Refer to the slides for more detail.

## 1.8 DCT

### 1.8.1 Motivation for DCT and iDCT

The major issue with using the DFT for image compression is border effects. Remember that when we split an image into patches, take the DFT of each patch, and reconstruct the image using only $k^2$ coefficients for each patch we are guaranteed to see border effects (unless $k^2 = N^2$). This happens because of the periodicity of the DFT. Remember that for a discrete signal $x$ we say that the DFT is periodic: $X(i) = X(i + N)$. The issue is that there's a big jump based on the original signal. $X(i)$ is most likely very different from $X(i + N - 1)$. If we think of the original signal as being periodic, there is a big difference between the first value we sample and the last value we sample in $[0, N - 1] \times [0, N - 1]$.

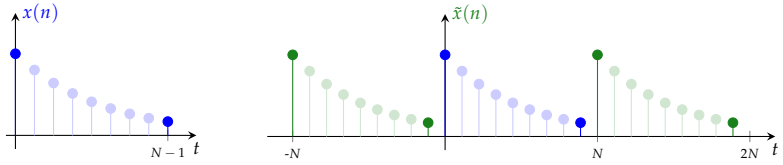


Figure 1.16: Periodic Extension of Input Signal

### 1.8.2 iDCT Definition

Luckily, there's a transform that is excellent for eliminating border effects called the inverse discrete cosine transform (iDCT). Notice that we have replaced the complex exponential we would expect to see in the iDFT with a cosine. There are no complex numbers involved in the iDCT. The original signal $x$ is still represented as a sum of oscillations, but the oscillations are cosines which help avoid border discontinuities.

$$\tilde{x}(n) := \frac{1}{\sqrt{N}} X(0) + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} X(k) \cos\left[\frac{\pi k(2n+1)}{2N}\right] \tag{1.33}$$

The iDCT yields an even extension. Put in plain english, the iDCT returns a version of the original signal that is symmetric around discontinuities. The figure below should be helpful. Refer to the slides for the complete derivation - it is very involved.

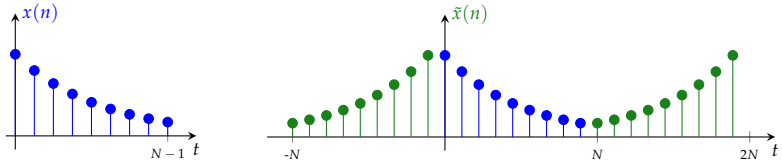


Figure 1.17: Even Extension of the iDCT

### 1.8.3 DCT

Of course we need a transform to put the original signal into a domain that is recoverable by the iDCT. This is of course the DCT. The definition is below. It assumes a real signal $x$ and the DCT output $X$ is also real. Notice that the normalization constants for $X(0)$ and $X(k \neq 0)$ are different. We write the DCT in parts to fully describe this behavior.

$$X(0) := \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi 0(2n+1)}{2N}\right] \qquad X(k) := \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi k(2n+1)}{2N}\right] \tag{1.34}$$

Note that the iDCT is an even function. If we put a mirror at $N - 1/2$, we can see this. Specifically, let's look at the samples at $n = N - 1$ and $n = N$.

First, the sample at $n = N - 1$:

$$\tilde{x}(N-1) := \frac{1}{\sqrt{N}}X(0) + \sqrt{\frac{2}{N}}\sum_{k=1}^{N-1} X(k)\cos\left[\frac{\pi k(N-1+1/2)}{N}\right] \tag{1.35}$$

$$= \frac{1}{\sqrt{N}}X(0) + \sqrt{\frac{2}{N}}\sum_{k=1}^{N-1} X(k)\cos\left[\pi k + \frac{\pi k(-1/2)}{N}\right] \tag{1.36}$$

Now, let's look at $n = N$:

$$\tilde{x}(N\quad) := \frac{1}{\sqrt{N}}X(0) + \sqrt{\frac{2}{N}}\sum_{k=1}^{N-1} X(k)\cos\left[\frac{\pi k(N\quad+1/2)}{N}\right] \tag{1.37}$$

$$= \frac{1}{\sqrt{N}}X(0) + \sqrt{\frac{2}{N}}\sum_{k=1}^{N-1} X(k)\cos\left[\pi k + \frac{\pi k(1/2)}{N}\right] \tag{1.38}$$

This process can be extended for any sample in the signal, as shown in the lecture slides. This is omitted here for the sake of concision. It is helpful to see this represented mathematically, although the visual representation in Figure **??** may be more intuitive.

### 1.8.4 The Even Extension of the iDCT

By now, we have seen that the iDCT is an even function due to the fact that its constituent parts, cosines, are even functions. We have seen empirically that the DCT reduces border effects in image reconstruction. Now we will see why.

We can formalize the iDCT's symmetry argument as follows:

$$\tilde{x}\Big[N + (n-1)\Big] = x\Big[N - n\Big] \tag{1.39}$$

To better visualize the symmetry, we can write this as:

$$\tilde{x}\Big[(N-1/2) + (n-1/2)\Big] = x\Big[(N-1/2) - (n-1/2)\Big] \tag{1.40}$$

What does this representation mean? It means that at the borders of the image, we no longer have the discontinuities that we have with the DFT. This is because the iDCT is even, whereas the DFT is not. This can be seen in Figure 1.17. With regards to image processing, we will see soon that this lets us avoid the nasty border effects that we encountered with the DFT.

### 1.8.5 DCT Basis

Let's return to the definition of the DCT, as defined above. Note that, unlike the DFT, there are *no* complex numbers involved in this transform. Both the signal and its transform are real. This will allow us to avoid the headaches associated with complex numbers that we have encountered thus far with the DFT.

Let's continue to compare the DCT with the DFT. We saw that the basis for the DFT is the complex exponential $e_{kN}$. That is, the DFT is the inner product of a signal with this

basis. We can write the DCT in the same way! However, the basis for the DCT is not the complex exponential $e_{kN}$, but rather $c_{kN}$, such that

$$c_{0N}(n) := \frac{1}{\sqrt{N}} \qquad c_{kN}(n) := \sqrt{\frac{2}{N}} \cos\left[\frac{\pi k(2n+1)}{2N}\right] \tag{1.41}$$

We can apply all of our usual intuition to this representation. That is, $X(k)$ tells us how much $x(n)$ resembles a discrete cosine oscillation with frequency $k$.

### 1.8.6 iDCT is the Inverse of the DCT

As the name would imply, the iDCT is, in fact, the inverse of the DCT. We won't go into the proof here, as you have already seen it several times with the DFT and other transforms we have covered up to now. That is,

$$\tilde{x} \equiv \mathcal{C}^{-1}(X) \equiv \mathcal{C}^{-1}(\mathcal{C}(x)) \equiv x \tag{1.42}$$

As we would imagine, Parseval's theorm also holds, which allows us to perform the reconstructions and error calculations that we have seen with the DFT.

## 1.9 2D Discrete Cosine Transform

Although it is helpful to conceptualize the DCT in 1D as an analog to the DFT, we will never be using the 1D DCT. We will only be using the 2D DCT, due to its superior performance in image processing. Our process of developing the 2D DCT will be similar to when we developed the 2D DFT.

Recall the definition of the 1D DCT:

$$X(0) := \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi 0(2n+1)}{2N}\right] \qquad X(k) := \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi k(2n+1)}{2N}\right] \tag{1.43}$$

In order to make the notation cleaner, let's introduce normalization constants $\nu_0 = 1/\sqrt{2}$ and $\nu_k = \sqrt{2}$ for $k \neq 0$. This is just to make the notation more compact when we move to 2D and have a heftier formula. Our 1D DCT can then be written as

$$X(k) := \frac{\nu_k}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi k(2n+1)}{2N}\right] \tag{1.44}$$

We can now proceed to define the 2D DFT as we did the 2D DFT. Most intuitively, that is as the vertical DCT of the horizontal DCTs (or vice versa). As a nested sum, this is

$$X(k,l) := \frac{\nu_k \nu_l}{N} \sum_{n=0}^{N-1} \left[ \sum_{m=0}^{N-1} x(m,n) \cos\left[\frac{\pi k(2m+1)}{2N}\right] \right] \cos\left[\frac{\pi l(2n+1)}{2N}\right] \tag{1.45}$$

More commonly, we write the 2D DCT as

$$X(k,l) := \frac{\nu_k \nu_l}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x(m,n) \cos\left[\frac{\pi\ k(2m+1)}{2N}\right] \cos\left[\frac{\pi l(2n+1)}{2N}\right] \tag{1.46}$$

Again, these normalization constants are the same as before, and are only to make the notation more concise. Our interpretation of the 2D DCT with respect to the 1D DCT is the same as our interpretation of the 2D DFT with respect to the 1D DFT. That is, it is a two dimensional extension obtained by taking the row-wise transforms of the transforms of the columns (or vice versa). Although the formula is a bit cumbersome, this should not be anything too extraordinary at this point.

### 1.9.1 The 2D DCT as an Inner Product

As we wrote the 2D DCT as an inner product with a 2D complex exponential, we can also write the 2D DCT as an inner product. To do this, we define a 2D discrete cosine with horizontal frequency $k$ and vertical frequency $l$ as follows

$$c_{klN}(n,m) := \frac{c_k}{\sqrt{N}} \cos\left[\frac{\pi k(2m+1)}{2N}\right] \frac{c_l}{\sqrt{N}} \cos\left[\frac{\pi l(2n+1)}{2N}\right] \tag{1.47}$$

Now we can rewrite the 2D DCT as an inner product with this discrete cosine. That is:

$$X(k,l) = \langle x, c_{klN} \rangle \tag{1.48}$$

This inner product representation carries with it its usual interpretation. That is, it tells us how similar a signal (image) is to a cosine oscillation with horizontal frequency $k$ and vertical frequency $l$. Note that orthonormality is retained as well. That is,

$$c_{klN}(n,m) = c_{kN} c_{lN} \tag{1.49}$$

### 1.9.2 2D iDCT

Before we construct the 2D iDCT, let's return to our definition of the 1D iDCT.

$$\tilde{x}(n) := \frac{1}{\sqrt{N}} X(0) + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} X(k) \cos\left[\frac{\pi k(2n+1)}{2N}\right] \tag{1.50}$$

Now, let's introduce our normalization constants again for the sake of concision. Let $\nu_0 = 1/\sqrt{2}$ and $\nu_k = 1$ for $k \neq 0$. *Note that these constants are the inverse of what they were before*. This should make sense, since, as we will see shortly, the 2D iDCT is the inverse of the 2D DCT!

$$\tilde{x}(n) := \sum_{k=1}^{N-1} \frac{\nu_k}{\sqrt{N}} X(k) \cos\left[\frac{\pi k(2n+1)}{2N}\right] \tag{1.51}$$

We can now develop the formula for the 2D DCT, as follows:

$$\tilde{x}(m,n) := \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \frac{\nu_k \nu_l}{N} X(k,l) \cos\left[\frac{\pi k(2m+1)}{2N}\right] \cos\left[\frac{\pi l(2n+1)}{2N}\right] \tag{1.52}$$

There is nothing new going on here - this inverse transform is just like all the others! There is, however, one key difference. As we have already seen, the iDCT is even symmetric. This is the case for the 2D iDCT two. Mathematically,

$$\tilde{x}\Big[(N-1/2)+(m-1/2),n\Big] = x\Big[(N-1/2)-(m-1/2),n\Big] \tag{1.53}$$

$$\tilde{x}\Big[m,(N-1/2)+(n-1/2)\Big] = x\Big[m,(N-1/2)-(n-1/2)\Big] \tag{1.54}$$

We have made a big deal of this symmetry, but why? Well, as we saw with the 1D iDCT, this symmetry means that there is no discontinuity between subsequent elements. This was, in general, not the case for the DFT. In the realm of image processing, this symmetry means that we will be able to avoid those nasty border effects that we got with the DFT when we partitioned the image into patches and transformed each patch. Now, the borders of each patch will "stitch" back together smoothly! This can be seen in the Lena reconstructions to follow.

### 1.9.3 The DCT is the Inverse of the DCT (Again!)

As we would expect with an abstraction from one dimension to two, nothing of importance changes. This is especially so with the most important fact regarding these transforms: invertibility. Like before, the iDCT is, in fact, the inverse of the DCT.

$$\tilde{x} \equiv \mathcal{C}^{-1}(X) \equiv \mathcal{C}^{-1}(\mathcal{C}(x)) \equiv x \tag{1.55}$$

It is important to note that equivalence here means that $\tilde{x}(n) = x(n)$ for $n \in [0, N-1]$. Otherwise, $\tilde{x}$ is an even extension of the original signal $x$. This is analogous to the case with the DFT in the first half of the course. We saw that the periodized spectrum of the DFT, when passed through the iDFT, returned a periodic extension of the original signal. This was not a problem, as we simply applied a low pass filter to avoid this issue. While we will not do the same thing here, it is important to be aware of this technicality, as it was then. Parseval's Theorem holds here as well, as we would expect.

Let's revisit our Lena image. If we compress and reconstruct our image using coefficients $0 \leq k, l \leq 10$, we now have a flawless reconstruction with *no border effects*. We have still compressed by a factor of 2.56, yet our error energy is only 0.26%! This means that you can upload an image to your computer, reduce the amount of storage it takes by 2.56 times, and still have a nearly perfect version of your image!

## 1.10 JPEG Image Compression

One particular application of the image processing techniques we've learned so far is JPEG compression, which is one of the most popular image compression schemes in use. We can imagine a color image, which has three channels: red, green, and blue. Each pixel is then a superposition of these three values, where are 8 bit integers (in other words, they take a value of 0 to 255), from which every color can be generated on a computer screen. If you've ever sneezed on your computer screen and seen red, green, and blue, this is


Figure 1.18: Reconstruction of Lena image. Indistinguishable from the original, yet with 2.56 less information stored!

what is happening - the water magnifies these pixels so that you can see the individual channels.

Although we are working with a grayscale image in the lab, it is useful to consider color images, as they are what this compression scheme is typically used on. A color image can be decomposed into its luminance and chrominance. The mechanics of this decomposition is beyond the scope of this course. Suffice it to say that our eye is much more sensitive to luminance than to chrominance.

To perform JPEG compression, we perform the 8 x 8 patchwise DCT that we have been doing thus far. However, we will now introduce importance quantization, which is a way of keeping the frequencies that our eyes are more sensitive to and getting rid of the frequencies that our eyes are less sensitive to. Mathematically, this is done by dividing each frequency in the patch $X(k,l)$ by some value $Q(k,l)$ such that if $Q(k,l)$ is close to 1, $X(k,l)$ does not change, but if $Q(k,l)$ is large, $X(k,l)$ becomes small, which requires less computer bits to encode. Rounding achieves the same purpose, as 1.0000 requires more information to store in a computer than 1, although they are mathematically equivalent.

Values of Q have been empirically determined to be the following:

$$\mathbf{Q}=\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 36 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

Similar to MP3 compression, JPEG uses psychosomatic factors in its compression scheme. That is, although we are discarding information by rounding by these seemingly arbitrary numbers, it is information that our eyes cannot perceive. Therefore, to our eyes, we are seeing the same image, but it requires less storage in a computer.

As we can see by the magnitude of the numbers, the low frequency information (top left corner) is generally more important (i.e., scaled down by lower numbers), and the high frequency information (bottom right corner) is generally less important (scaled down by higher numbers). This ties back to what we learned way back in one-dimensional signal processing about low and high frequencies. Cool!