Contents

1 Signals, images and sequences

In the previous chapter, we made a simple computer vision system. We broke the image into edges and labeled them to make progress in interpreting the 3d shape of a scene. But we had to work in a very constrained world so that our brittle processing steps would give useful information about edges and their labels.

Of course we want to build a vision system that operates in the real world. One such system is the human visual system. Although much remains to be understood about how our visual system processes images, we have a fairly good idea of what happens at the initial stages of visual processing, and it will turn out to be similar to some of the filtering we discuss in this chapter. While we're inspired by the biology, here we describe some mathematically simple processing that will help us to parse an image into useful tokens, low-level features that will be useful later to construct visual interpretations.

We'd like for our processing to enhance image structures of use for subsequent interpretation, and to remove variability within the image that makes more difficult comparisons with previously learned visual signals. Let's proceed by invoking the simplest mathematical processing we can think of, and see how far it takes us toward these goals.

Linear filters are one of the main tools to perform computations in images. Linear filters as computing structures for vision have received a lot of attention because of their surprising success in modeling some aspect of the processing carried our by early visual areas such as the retina, the lateral geniculate nucleus (LGN) and primary visual cortex (V1). Neurons are modeled by a linear first stage that takes a weighted linear combination of the input image and a second stage which performs a non-linearity. This model is certainly a simplification but it is remarkably successful at explaining a large number of observations in the primate visual system.

Many of the filters that we will review in this chapter and the next one have become popular because of their functionality as image processing tools and/or because of their fit to observations in the early visual system.

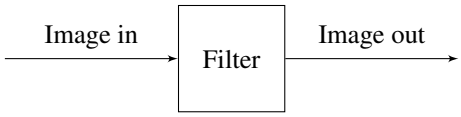


Figure 1.1
Generic image processing module.

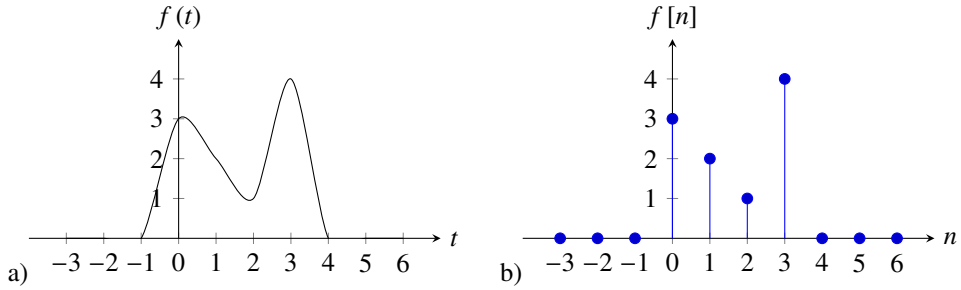


Figure 1.2
a) A continuous signal, and b) a discrete signal.

1.1 Signals and systems

In this chapter we will introduce some basic tools to characterize images and simple image processing systems. We will describe tools from signal and image processing. For a deeper understanding there are many books [] devoted to signal processing, providing essential tools for any computer vision scientist. We will present signal processing tools from a computer vision perspective. Our goal will be to extract from images information useful to build meaningful representations of the image in order to understand its content.

The kind of systems that we will study here are of the kind shown in figure 1.7 that take an image as input, perform some filtering operation, and output another image. In general a filter will be used to remove some image components or enhance others.

The tools we will review here allow representing signals (1D sequences, images or movies) and the filters used to process them.

1.2 Signals, images and sequences

1.2.1 Continuous and discrete signals

If we consider the brightness captured by one photo sensor, we could write it as a one dimensional function of time: $f(t)$, where $f(t)$ denotes the measured brightness value at time t, and t is a continuous variable that can take on any real value.

Although most of the signals that exist in nature are continuous signals, when we introduce them into a computer they are sampled and transformed into discrete signals.

The function $f(t)$ can be sampled in time (as is done in a video) where the values are only captured at discrete times (e.g., 30 times per second). In that case, the function f will be defined only on discrete time instants and we will write the sequence of measured values as: $f[n]$, where n can only take on discrete values. The relationship between the discrete values and the continuous function is given by the sampling:

$$f[n] = f(n\,\Delta T) \tag{1.1}$$

where $\Delta\, T$ is the sampling period. For instance, $\Delta T = 1/30$ secs in the case of sampling the signal 30 times per second.

As is common in signal processing, we will use parenthesis to indicate continuous variables and brackets to denote discrete variables. For instance, if we sample the signal shown in figure 1.2.a once every second ($\Delta T = 1$ secs) we get the discrete signal shown in figure 1.2.b.

The signal in figure 1.2.b is a function that takes on the values $f[0] = 3$, $f[1] = 2$, $f[2] = 1$ and $f[3] = 4$ and all other values of $f[n] = 0$. In most of the book we will work with discrete signals. In many cases it will be convenient to write discrete signals as vectors. Using vector notation we will write the previous signal as a column vector $f = [3, 2, 1, 4]^T$, where T denotes transpose.

Images and sequences are also discrete signals. Gray scale images are two dimensional signals that can be encoded as arrays of pixels: $\mathbf{I}[n, m]$. Sequences are three dimensional and we will write them as $\mathbf{I}[n, m, t]$. Figure 1.3 shows an image of size 18×18 pixels plotted as a 2D signal. Can you guess the object that appears in this image?

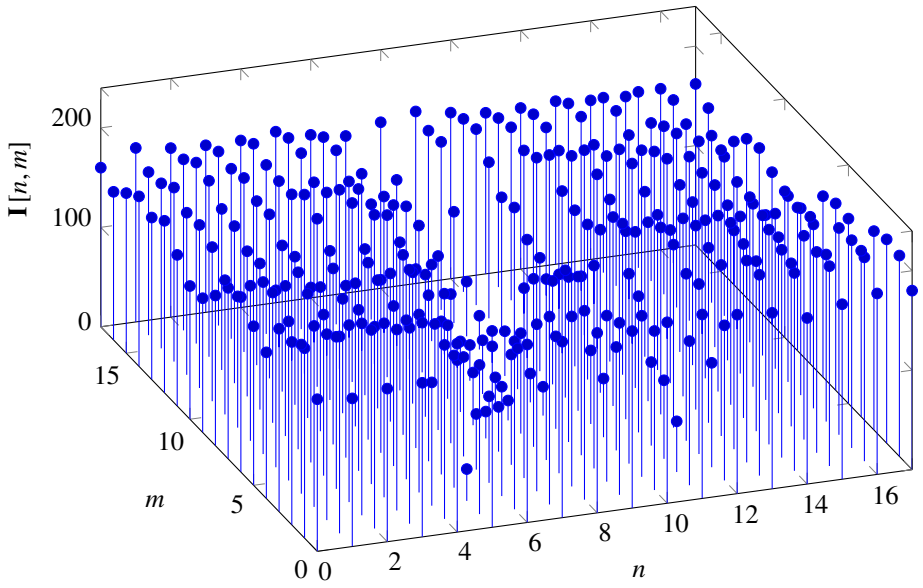


Figure 1.3
A discrete 2D signal.

Similarly to 1D signals, we can write this 2D signal as a matrix:

$$\mathbf{I} = \begin{bmatrix}
160 & 175 & 171 & 168 & 168 & 172 & 164 & 158 & 167 & 173 & 167 & 163 & 162 & 164 & 160 & 159 & 163 & 162 \\
149 & 164 & 172 & 175 & 178 & 179 & 176 & 118 & 97 & 168 & 175 & 171 & 169 & 175 & 176 & 177 & 165 & 152 \\
161 & 166 & 182 & 171 & 170 & 177 & 175 & 116 & 109 & 169 & 177 & 173 & 168 & 175 & 175 & 159 & 153 & 123 \\
171 & 174 & 177 & 175 & 167 & 161 & 157 & 138 & 103 & 112 & 157 & 164 & 159 & 160 & 165 & 169 & 148 & 144 \\
163 & 163 & 162 & 165 & 167 & 164 & 178 & 167 & 77 & 55 & 134 & 170 & 167 & 162 & 164 & 175 & 168 & 160 \\
173 & 164 & 158 & 165 & 180 & 180 & 150 & 89 & 61 & 34 & 137 & 186 & 186 & 182 & 175 & 165 & 160 & 164 \\
152 & 155 & 146 & 147 & 169 & 180 & 163 & 51 & 24 & 32 & 119 & 163 & 175 & 182 & 181 & 162 & 148 & 153 \\
134 & 135 & 147 & 149 & 150 & 147 & 148 & 62 & 36 & 46 & 114 & 157 & 163 & 167 & 169 & 163 & 146 & 147 \\
135 & 132 & 131 & 125 & 115 & 129 & 132 & 74 & 54 & 41 & 104 & 156 & 152 & 156 & 164 & 156 & 141 & 144 \\
151 & 155 & 151 & 145 & 144 & 149 & 143 & 71 & 31 & 29 & 129 & 164 & 157 & 155 & 159 & 158 & 156 & 148 \\
172 & 174 & 178 & 177 & 177 & 181 & 174 & 54 & 21 & 29 & 136 & 190 & 180 & 179 & 176 & 184 & 187 & 182 \\
177 & 178 & 176 & 173 & 174 & 180 & 150 & 27 & 101 & 94 & 74 & 189 & 188 & 186 & 183 & 186 & 188 & 187 \\
160 & 160 & 163 & 163 & 161 & 167 & 100 & 45 & 169 & 166 & 59 & 136 & 184 & 176 & 175 & 177 & 185 & 186 \\
147 & 150 & 153 & 155 & 160 & 155 & 56 & 111 & 182 & 180 & 104 & 84 & 168 & 172 & 171 & 164 & 168 & 167 \\
184 & 182 & 178 & 175 & 179 & 133 & 86 & 191 & 201 & 204 & 191 & 79 & 172 & 220 & 217 & 205 & 209 & 200 \\
184 & 187 & 192 & 182 & 124 & 32 & 109 & 168 & 171 & 167 & 163 & 51 & 105 & 203 & 209 & 203 & 210 & 205 \\
191 & 198 & 203 & 197 & 175 & 149 & 169 & 189 & 190 & 173 & 160 & 145 & 156 & 202 & 199 & 201 & 205 & 202 \\
153 & 149 & 153 & 155 & 173 & 182 & 179 & 177 & 182 & 177 & 182 & 185 & 179 & 177 & 167 & 176 & 182 & 180
\end{bmatrix}$$

This matrix represents the same image of 18×18 pixels as in figure 1.15. Again, it is hard to grasp the content of the image by just looking at this matrix. Figure 1.4 visualizes the matrix as an image where each value is shown as a different gray-scale value.

It is also convenient to encode images as a 1D vector by concatenating all the image columns into a long column vector of $M \times N$ values.

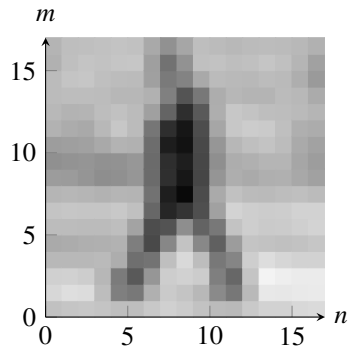


Figure 1.4
Gray scale image showing a person walking in the street. This tiny image has only 18×18 pixels.

Although in practice images will always be discrete signals, in some cases it is useful to work on the continuous domain as it simplifies the derivation of analytical solutions. This was the case in the previous chapter were we used the image gradient in the continuous domain and then approximated it in the discrete space domain. In those cases we will write images as $\mathbf{I}(x, y)$ and sequences as $\mathbf{I}(x, y, t)$.

1.2.2 Signal space

As we have done already several times, it is very useful to think of signals and images as vectors. Let us define the two finite length column vectors f and g of length N. f and g can be the sequence of numbers of a 1D signal or the concatenation of all the columns (or rows) of a 2D image into a 1D column vector. Although we have only worked with real signals and images, later we will manipulate also signals and images made of complex numbers.

Then we can define:

The scalar product between two (complex) signals is:

$$\langle f, g \rangle = \sum_{n=0}^{N-1} f[n]\, g^*[n] = f^T g^* \tag{1.2}$$

where * denotes complex conjugate. Two signals are orthogonal if $\langle f, g \rangle = 0$.

The scalar product also induces a norm. The squared L2 norm of a signal is:

$$E_f = \|f\|^2 = \langle f, f \rangle = \sum_{n=0}^{N-1} |f[n]|^2 = f^T f^* \tag{1.3}$$

some times this is referred to as the energy of the signal.

If we want to compare two signals, we can use the squared L2 distance:

$$d^2_{f,g} = \|f - g\|^2 = \sum_{n=0}^{N-1} |f[n] - g[n]|^2 = E_f + E_g - 2\langle f, g \rangle \tag{1.4}$$

The set of signals with finite L2 norm form a Hilbert space. We can also define basis, other metrics, ...

The same applies to continuous functions.

1.2.3 Properties of a signal

Signals and images are very high dimensional sequences of numbers. One of our goals is to find representations that extract from the signal some basic quantities that allow us making explicit signal properties that might allow us to understand its content. Here are some important quantities:

DC value is the mean signal value[1]:

$$DC = \sum_n f\,[n] \tag{1.5}$$

The energy of the signal is defined as the squared L2 norm of the signal:

$$E = \|f\|^2 = \sum_n |f\,[n]\,|^2 \tag{1.6}$$

For finite energy signals, there are other quantities of interest:

Center of mass of the signal measure the average position where the signal variations are located:

$$\mu = \frac{1}{E} \sum_n n\,|f\,[n]\,|^2 \tag{1.7}$$

The variance in the spatial domain measures how compact is the signal:

$$\sigma^2 = \frac{1}{E} \sum_n (n-\mu)^2\,|f\,[n]\,|^2 \tag{1.8}$$

For an image, the mean is a 2D vector and the co-variance is a 2×2 matrix.

1.3 Linear Filtering

Figure 1.5 we show what a general filter looks like. It takes as input a signal g and outputs the signal f such that

$$f = H(g) \tag{1.9}$$

where H is an arbitrary function.

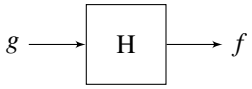


Figure 1.5
Notation to describe a general non-linear filter that takes as input a signal g and outputs a signal f so that $f = H(g)$.

This transformation is very general and it can do all sorts of complex things. For instance, it could detect edges in images, recognize objects, detect motion in sequences or apply aesthetic transformations to a picture. This generality makes it also very difficult to characterize them. Therefore, we will start with simpler family of filters. Among all

[1] DC means *direct current* and it comes from electric engineering. Although most signals have nothing to do with currents, the term DC is still commonly used.

possible filters, the simplest ones are linear filters. They are nice because there are powerful tools to describe and implement them. On the other hand they represent a very small portion of all the possible filters one could implement. Despite this limitation we will show in the next chapter how some simple linear filters are capable of creating very interesting applications.

To make things more concrete, lets assume the input is a 1D signal with length N that we will write as $g[n]$, and the output is another 1D signal with length M that we will write as $f[n]$. Most of the times we will work with input and output pairs with the same length $M = N$. A linear filter, in its most general form, can be written as:

$$f[n] = \sum_{k=0}^{N-1} h[n,k]\, g[k] \quad for\ n \in [0, M-1] \tag{1.10}$$

where each output value $f[n]$ is a linear combination of values of the input signal $g[n]$ with weights $h[n,k]$. To help to visualize the operation perform by the linear filter it is useful to write it in matrix form:

$$\begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M-1] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[0,1] & ... & h[0,N-1] \\ h[1,0] & h[1,1] & ... & h[1,N-1] \\ \vdots & \vdots & \vdots & \vdots \\ h[M-1,0] & h[M-1,1] & ... & h[M-1,N-1] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[N-1] \end{bmatrix} \tag{1.11}$$

which we will write as

$$f = Hg \tag{1.12}$$

The matrix H will have size $M \times N$ where N is the length of the input signal $g[n]$ and M is the length of the output signal $f[n]$. We will use the matrix formulation many times in this book.

In 2D dimensions each pixel of the output image is replaced by a linear combination of pixels of the input image. If horizontal and vertical positions are indexed by n and m, the output image is $f[n,m]$, and the input image is $g[n,m]$, then a general linear filtering of the image is

$$f[n,m] = \sum_{k,\, l=0}^{N-1,\, M-1} h[n,m,k,l]\, g[k,l] \tag{1.13}$$

By writing the images as column vectors, concatenating all the image columns into a long vector, we can also write the previous equation as a matrix times a vector.

1.3.1 Convolution and translation invariant filtering

One particular case of a linear filter is a translation invariant linear filter.

Typically, we don't know where within the image we expect to find any given item (Fig. 1.6), so we often want to process the image in a spatially invariant manner, the

Figure 1.6
A fundamental property of images is translation invariance–the same image may appear at arbitrary spatial positions within the image. Image credit: Fredo Durand.

same processing algorithm at every pixel. In that case, the processing becomes a *linear convolution* of the image data with some filter.

Let's start with the 1D case. As example of a translation invariant filter is a filter for which the output for the sample n is twice the value of the input at that same time minus the sum of the two previous time steps. This is:

$$\begin{aligned} f[0] &= 2g[0] - g[-1] - g[-2] \\ f[1] &= 2g[1] - g[0] - g[-1] \\ f[2] &= 2g[2] - g[1] - g[0] \\ &\dots \\ f[n] &= 2g[n] - g[n-1] - g[n-2] \end{aligned} \tag{1.14}$$

A filter is linear translation invariant (LTI) if it is linear and when we translate the input signal by m samples, the output is also translated by m samples. This means that the behavior of the filter does not change depending on the location of the input.

Linear translation invariance imposes a strong constraint on the form of equation 1.10. The weighting, h, for the linear combination of the input image pixels, g, is only a function of the spatial offset from the pixels of g. For a 1D signal, a linear convolution, denoted $\circ$, of h and g is:

$$f[n] = h \circ g = \sum_{k=0}^{N-1} h[n-k]\, g[k] \tag{1.15}$$

for the previous example $h = [2, -1, -1]$. N is the length of the signal $g[n]$ and we assume it is zero outside of the interval $n \in [0, N-1]$.

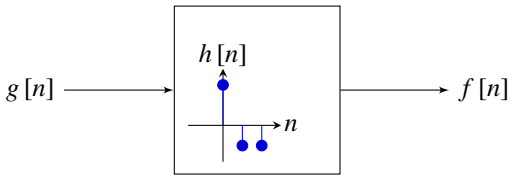


Figure 1.7
Example of a linear translation invariant signal processing module. The function inside the box correspond to the convolution kernel.

Most of the filters we will use will have what is called finite support. A filter has finite support if the filter kernel h has non-zero values only at a finite number of locations. In the signal processing literature those filters are called finite impulse response filters (FIR).

In the 1D case, it helps to make explicit the structure of the matrix:

$$\begin{bmatrix} f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[N-1] \end{bmatrix} = \begin{bmatrix} h[0] & h[-1] & h[-2] & ... & h[1-N] \\ h[1] & h[0] & h[-1] & ... & h[2-N] \\ h[2] & h[1] & h[0] & ... & h[3-N] \\ \vdots & \vdots & \vdots & \vdots & \\ h[N-1] & h[N-2] & h[N-3] & ... & h[0] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ g[2] \\ \vdots \\ g[N-1] \end{bmatrix} \tag{1.16}$$

Figure 1.8 shows the convolution of a kernel, h, with a 1D signal, g. h and g are shown in the top row. Subsequent rows show the implementation. $h[m-k]$ is just h, offset by m pixels and reversed. We multiply this term-by-term with g and sum those weighted values of g[m] to form the output signal, f[m].

In two dimensions, the processing is analogous: The input filter is flipped vertically and horizontally, then slid over the image to record the inner product with the image everywhere. Mathematically, this is:

$$f[m,n] = h \circ g = \sum_{k,l} h[m-k, n-l]\, g[k,l] \tag{1.17}$$

Figure 1.9 shows the 2D convolution of a kernel h with an image, g. The particular kernel used in the figure averages in the vertical direction and takes differences horizontally. The output image reflects that processing, with horizontal differences accentuated and vertical changes diminished.

Figure 1.10 shows several simple convolution examples. Figure 1.10.a shows a kernel with a single central non-zero element, convolved with any image, gives back that same image (even at the boundaries, by the way, since any pixels beyond the boundaries are multiplied by zero). This kernel is called the impulse and we will discuss it later. Figure 1.10.b shows a kernel that produces a shift of the input image. For the last example shown in figure 1.10.c, can you guess what linear convolution will cause the image to rotate?

At the center of rotation, the center pixel should be output, no matter what the surrounding pixels are, so that can only be implemented by convolution with an impulse. But at the top left corner, one wants to grab a pixel from, say, 5 pixels down and to the right, and from the bottom one needs to grab the pixel from about 5 pixels up and to the right. So this rotation operation can't be written as a spatially invariant convolution.

1.3.1.1 Properties of the convolution The convolution is a linear operation that will be extensively used thorough the book and it is important to be familiar with some of its properties.

- Using equation. 1.15 or equation 1.17 is it easy to show that convolution is commutative operator:

$$h[n] \circ g[n] = g[n] \circ h[n] \tag{1.18}$$

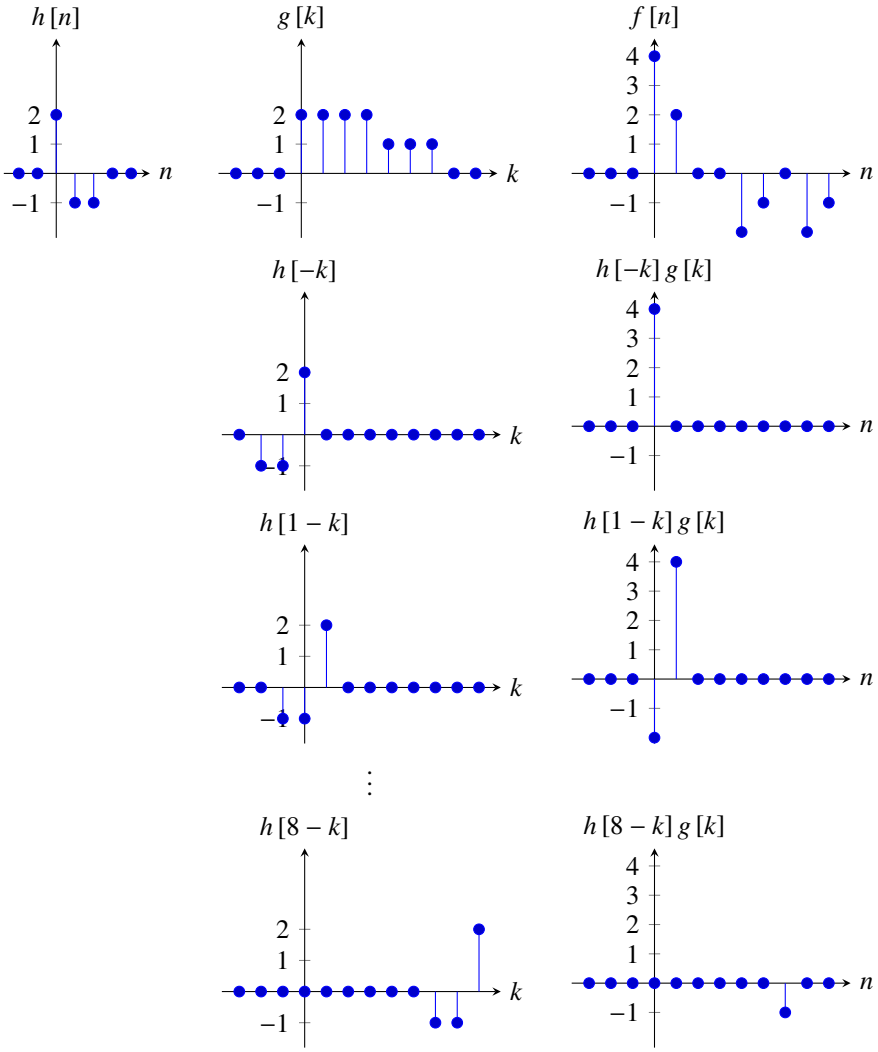


Figure 1.8
Illustration in 1-d of the steps in computing the convolution of a kernel h with a signal g. Shifted and offset versions of the kernel h provide the weights to construct f[m] from a linear combination of the samples of g[m].

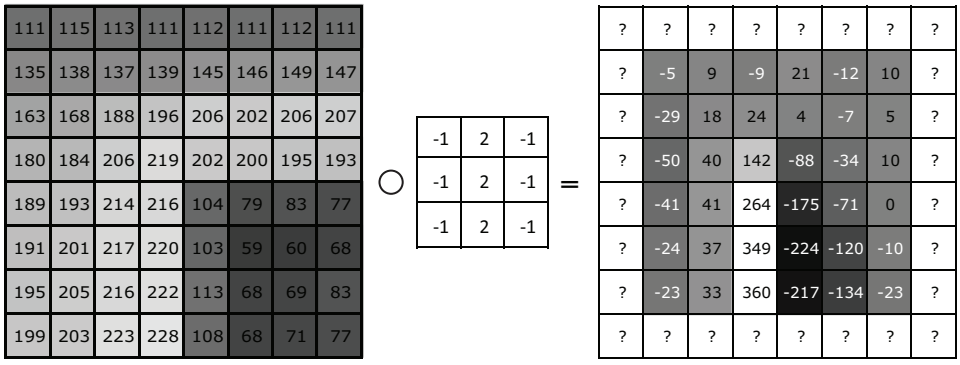


Figure 1.9

Illustration of a 2-d convolution of an input image, g, convolved with a kernel, h, giving the output image, f. The images are shown with both their pixel values and the corresponding image intensities (the assignment of intensities to numbers was rescaled for the output image, f). Border pixel values of the output image are not determined by the convolution, since the kernel would include pixel values outside of the input image.

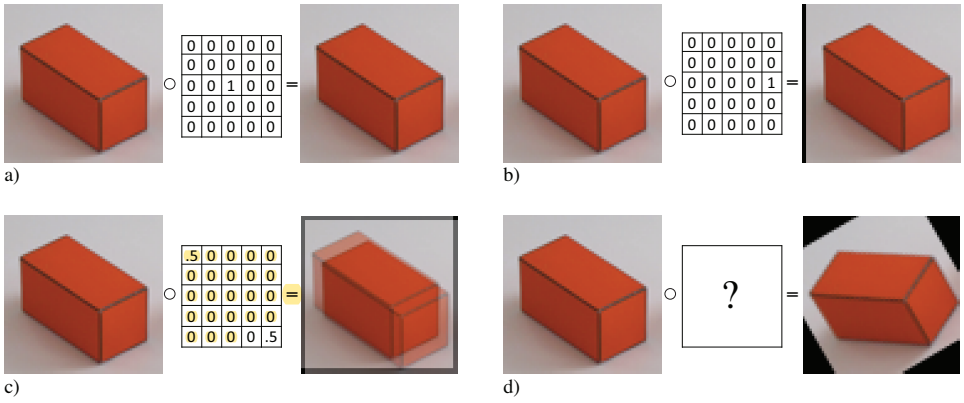


Figure 1.10

a) An impulse convolved with the input image gives no change (each color channel is convolved with the same kernel). b) A shifted impulse shifts the image. c) Sum of two shifted copies of the image. d) The text discusses why there is no space invariant convolution kernel can rotate an image. All the examples use zero padding for handling boundary conditions.

this means that the order of convolutions is irrelevant.

- It is associative:

$$h[n] \circ g[n] \circ q[n] = h[n] \circ (g[n] \circ q[n]) = (h[n] \circ g[n]) \circ q[n] \tag{1.19}$$

- It is distributive with respect to the sum:

$$h[n] \circ (f[n] + g[n]) = h[n] \circ f[n] + h[n] \circ g[n] \tag{1.20}$$

- Another interesting property involves reversing the shifts between the two convolved functions. If $f[n] = h[n] \circ g[n]$, then:

$$f[n - n_0] = h[n] \circ g[n - n_0] = h[n - n_0] \circ g[n] \tag{1.21}$$

- The convolution of a signal a support of N samples with another one with a support of M samples results in a signal with a support $L \leq M + N - 1$.
- The convolution also has an identity function which we will introduce in section 1.4.1.

1.3.1.2 Handling boundaries When implementing a convolution, one is confronted with the question of what to do at the image boundaries. There's really no satisfactory answer for how to handle the boundaries that works well for all applications. One solution consists in omitting from the output any pixels that are affected by the input boundary. The issue with this is that the output will have a different size than the input and, for large convolutional kernels, there might be a large portion of the output image missing.

The most general approach consists in extending the input image by adding additional pixels so that the output can have the same size as the input. So, for a kernel with support $[-N, N] \times [-M, M]$, one has to add $N/2$ additional pixels left and right of the image and $M/2$ pixels at the top and bottom. Then, the output will have the same size as the original input image.

Some typical choices for how to pad the input image are (see fig. 1.11):

- Zero padding: set the pixels outside the boundary to zero (or to some other constant such as the mean image value).
- Repeat padding: set the value to that of the nearest output image pixel with valid mask inputs.
- Mirror padding: reflect the valid image pixels over the boundary of valid output pixels. This is the most common approach and the one that gives the best results.
- Circular padding: extend the image by replicating the pixels from the other size. If the image has size $P \times Q$, then, circular padding consists in: $\mathbf{I}[n, m] = \mathbf{I}[mod(n, P), mod(m, Q)]$. This padding transform the finite length signal into a periodic infinite length signal. Although this will introduce many artifacts, it is a convenient extension for analytical derivations.

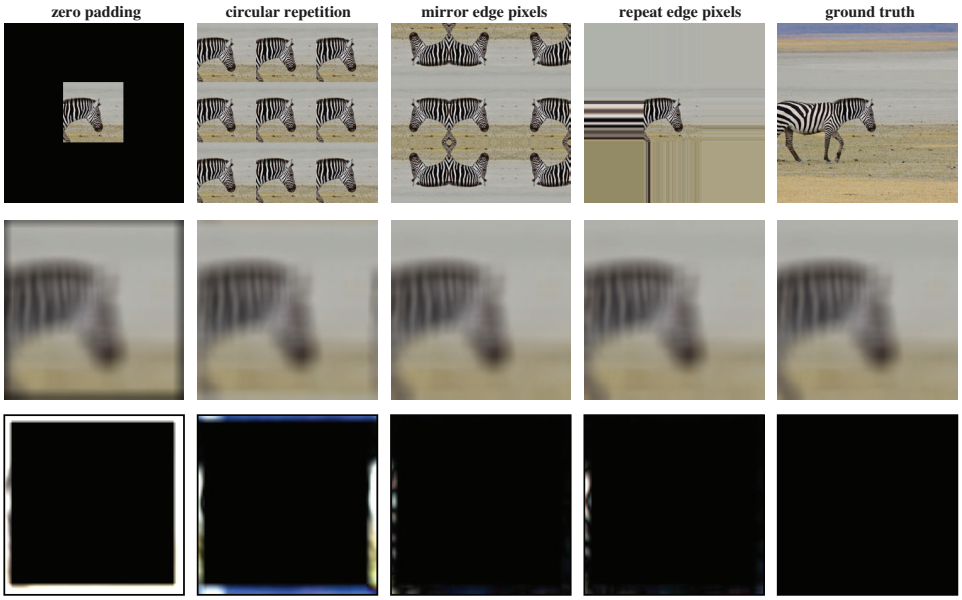


Figure 1.11
Boundary extensions are different ways of approximating the ground truth image that exists beyond the image boundary. Each column shows: a) Different types of boundary extension. The last image shows the ground truth. b) the output of convolving the image with a kernel that is a box of 1 with size 11×11. The output only shows the central region that corresponds to the input image without boundary extension. c) difference between each output and the ground truth output, last column of (b). Note that the ground truth will not be available in practice.

1.3.1.3 Convolution in the continuous domain Although in practice all the convolutions will be done in the discrete domain, the continuous domain is important for analytical derivations and for characterizing signals before they are captured by the camera.

The convolution in the continuous domain is defined in the same way as the discrete equations but replacing the sums by integrals. Given two continuous signals $h(t)$ and $g(t)$, the convolution operator is defined as:

$$f(t) = h(t) \circ g(t) = \int_{-\infty}^{\infty} h(t-\tau) g(\tau) d\tau \tag{1.22}$$

The properties of the continuous domain convolution are analogous to the properties of the discrete convolution, with the commutative, associative and distributive relationships still holding.

1.3.2 Correlation and template matching

Another form of writing a translation invariant filter is using the correlation operator. The correlation provides a simple technique to locate a template in an image.

1.3.2.1 Correlation vs. convolution The correlation and the convolution are closely related. The convolution between image g and filter h is:

$$f[m,n] = h \circ g = \sum_{k,l} h[m-k, n-l]\, g[k,l] \tag{1.23}$$

where the sum is done over the support of the filter h. The correlation between the image g and the filter h is written as:

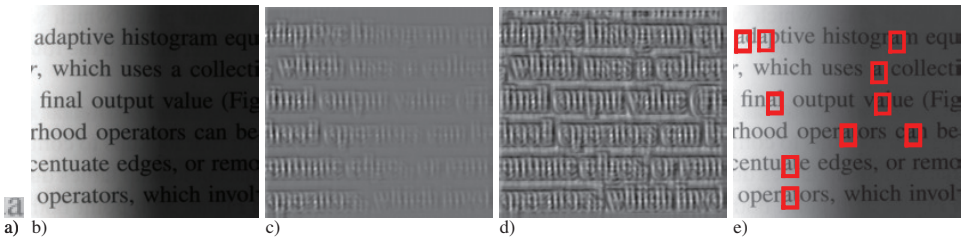
$$f[m,n] = h * g = \sum_{k,l} h[m+k, n+l]\, g[k,l] \tag{1.24}$$

In the correlation, the filter is not inverted left-right and up-down as it is done in the convolution. In particular, note that the correlation and convolution operators are identical when the filter h is symmetric.

The difference between the two operators is that the convolution is commutative and associative while the correlation is not. The correlation breaks the symmetry between the two functions h and g. For instance, in the correlation, shifting h is not equivalent to shifting g.

1.3.2.2 Template matching and normalized correlation Template matching can be defined as detection of complex patterns such as objects within cluttered signals such as images. For instance figure 1.12 illustrates the detection of the "a" letters through matching an example "a" template (figure 1.12(a)) in a given text image (figure 1.12(b)).

Although correlation is a potential method for template matching it also has certain flaws. Consider matching the "a" template in figure 1.12(a) in a given input image. Just by increasing the brightness of the image in different parts we can increase the filter response



a) b) c) d) e)

Figure 1.12
a) Template. b) Input image. c) Output of the correlation between the image and the template. d) Output of the normalized correlation. e) Locations with values of the normalized correlation above 75% of its maximum value.

since correlation is essentially a multiplication between the filter f and any input image patch g, and note that all the values are positive in f and g. This suggests that in bright white regions we will have the maximum responses of the template. One way of improving the robustness of the template f would be introducing negative values inside it by constructing a zero-mean template:

$$f' = f - m_f \quad \text{where} \quad m_f = \frac{1}{MN} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \tag{1.25}$$

To further improve the robustness we can normalize both the filter f and the applied image patch g with standard deviations, eventually obtaining normalized cross-correlation (NCC) as:

$$\begin{aligned} NCC(f,g) = \left\langle \bar{f}, \bar{g} \right\rangle \quad \text{where} \quad \bar{f} &= \frac{f - m_f}{\sqrt{\left\langle f - m_f, f - m_f \right\rangle}} = \frac{f'}{\sqrt{\langle f, f' \rangle}}, \\ \bar{g} &= \frac{g - m_g}{\sqrt{\left\langle g - m_g, g - m_g \right\rangle}} = \frac{g'}{\sqrt{\langle f', f' \rangle}} \end{aligned} \tag{1.26}$$

Note that both $\bar{f}$ and $\bar{g}$ are unit norm vectors. Therefore $NCC(f,g) = \left\langle \bar{f}, \bar{g} \right\rangle = \|\bar{f}\| \, \|\bar{g}\| \, cos\alpha = cos\alpha$ where α is the angle between the vectors f' and g'.

Another common method of comparing patches is through sum of squared distances (SSD), also referred to as squared L2 distance:

$$SSD(f,g) = \|f - g\|^2 = \sum_n |f[n] - g[n]|^2 = E_f + E_g - 2\langle f, g \rangle \tag{1.27}$$

However this method also suffers in extreme illumination changes as the distance is strongly effected by the energy of the signals E_f and E_g. We can remove such undesired effects through L2 normalization of signals $\hat{f} = \frac{f}{\|f\|}$ and $\hat{g} = \frac{g}{\|g\|}$, and eventually obtain normalized

squared L2 distance:

$$SSD(\hat{f},\hat{g}) = \|\hat{f}-\hat{g}\|^2 = E_{\hat{f}} + E_{\hat{g}} - 2\langle f,g\rangle = 2 - 2\left\langle \hat{f},\hat{g}\right\rangle \tag{1.28}$$

where $E_{\hat{f}} = E_{\hat{g}} = 1$ as $\hat{f}$ and $\hat{g}$ are unit norm vectors due to the normalization. An important factor to note here is that the distance is no longer effected by the norm of the signals, but it is merely defined by the angle between two signals as follows:

$$SSD(\hat{f},\hat{g}) = 2 - 2cos\theta \quad \text{since} \quad \left\langle \hat{f},\hat{g}\right\rangle = \|\hat{f}\|\,\|\hat{g}\|\,cos\theta = cos\theta \tag{1.29}$$

where θ is the angle between the vectors f and g. The quantity $\left\langle \hat{f},\hat{g}\right\rangle$ is also referred to as cosine similarity, a well known similarity metric that is essentially the cosine of the angle between two vectors defined as follows:

$$cos_{sim}(f,g) = \frac{\langle f,g\rangle}{\|f\|\,\|g\|} \tag{1.30}$$

Note that cosine similarity is bounded by the interval $[-1,+1]$ as $cos\theta$ is. Hence $SSD(\hat{f},\hat{g})$ is bounded by the interval $[0,4]$.

If we want to have a measure of similarity between two signals that is invariant to overall image brightness, then a more appropriate measure is the angle. Note that both NCC and cos_{sim} are angular similarity measures and they are not effected by the energy of the signals such as illumination changes in images. The major difference between them is that θ in cos_{sim} is the angle between original signals f and g whereas α in NCC is the angle between the zero-mean vectors f' and g' defined in (1.25).

Convolution and correlation operators are the main building blocks of the convolutional neural networks which will be discussed in chapter **??**.

1.4 Special signals

There are two families of signals that deserve special attention due to their role in modeling linear systems: the impulse and sinusoidal waves.

1.4.1 The impulse

One important discrete signal is the impulse (also called the Kronecker delta function). The impulse in 1D is the signal defined as:

$$\delta[n] = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \neq 0 \end{cases} \tag{1.31}$$

it takes the value 1 for $n = 0$ and it is zero everywhere else.

This signal is important because it has some very useful properties. The most important one is that it behaves as the identity function for the convolution. The result of convolving

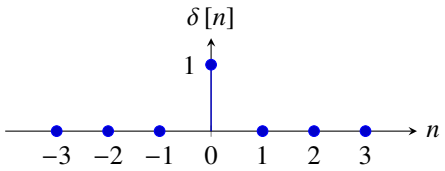


Figure 1.13
Impulse signal.

a signal $g[n]$ with the impulse signal is the same signal:

$$f[n] = \delta \circ g = \sum_k \delta[n-k]\, g[k] = g[n] \tag{1.32}$$

Convolving a signal f with a translated impulse $\delta[n-n_0]$ results in a translated signal:

$$f[n-n_0] = \delta[n-n_0] \circ f[n] \tag{1.33}$$

For an LTI filter with kernel $h[n]$, the output from an impulse is $f[n] = h[n]$. The output of a translated impulse $\delta[n-n_0]$ is $h[n-n_0]$. This is why the filter kernel $h[n]$ is also called the impulse response of the system. For an unknown system, one way of finding the convolution kernel h consists in computing the output of the system when the input is an impulse.

If you are in a room and you clap (which is a good approximation to an impulse of sound), the echoes that you hear are very close to the impulse response of the system formed by the acoustics of the room. Any sounds that originates at the location of your clap will produce a sound in the room that will be qualitatively similar to the convolution of the acoustic signal with the echoes that you hear before when you clapped.

To see how the shift operation works, let's see what happens in 1D. The linear system that shifts the signal by one sample can be written as $h[n] = \delta[n-1]$. In matrix form this is:

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ g[2] \\ \vdots \\ g[N-1] \end{bmatrix} = \begin{bmatrix} 0 \\ g[0] \\ g[1] \\ \vdots \\ g[N-2] \end{bmatrix} \tag{1.34}$$

The first zero is a boundary artifact (in this example we are using zero-padding to extend the signal). The rest of the signal is translated by 1 location.

The set of impulse signals, $\delta[n-k]$, delayed for all values $k \in [0, N-1]$, constitutes a basis for the set of discrete signals defined in the domain $[0, N-1]$. Any discrete signal

can be written as a linear combination of delayed impulse responses:

$$f[n] = \sum_{k=0}^{N-1} f[k]\,\delta[n-k] \tag{1.35}$$

where the values $f[k]$ are the coefficients used for the linear combination of the set of impulses.

In 2D, the impulse is

$$\delta[n,m] = \begin{cases} 1 & \text{if } n = 0,\ m = 0 \\ 0 & \text{either } n \neq 0 \text{ or } m \neq 0 \end{cases} \tag{1.36}$$

The 2D impulse is a separable function, meaning that it can be written as the product of two 1D functions, both impulses:

$$\delta[n,m] = \delta[n]\,\delta[m] \tag{1.37}$$

Convolution by an impulse can be used to translate an image:

$$f[n-n_0, m-m_0] = \delta[n-n_0, m-m_0] \circ f[n,m] \tag{1.38}$$

Although this might seem like an obscure way of doing such a simple image operation, it is very useful for analytical derivations.

The impulse can also be defined on the continuous domain. The continuous impulse is represented as $\delta(t)$ with $t \in \Re$. The impulse function is defined as being zero everywhere but at the origin where it takes an infinitely high value so that its integral is equal to 1:

$$\int_{-\infty}^{\infty} \delta(t)dt = 1 \tag{1.39}$$

The impulse function is also called the impulse distribution (as it is not a function in the strict sense) or the Dirac delta function.

The impulse has several important properties:

- Scaling property: $\delta(at) = \delta(t)/|a|$
- Symmetry: $\delta(-t) = \delta(t)$
- Sampling property: $f(t)\delta(t-a) = f(a)\delta(t-a)$ where a is a constant. From this, we have:

$$\int_{-\infty}^{\infty} f(t)\delta(t-a)dt = f(a) \tag{1.40}$$

- As in the discrete case, the continuous impulse is also the identity for the convolution: $f(t) \circ \delta(t) = f(t)$.

This function will be useful in analytical derivations and we will see it come back in section 1.6 when we discuss sampling.

1.4.2 Cosine and sine waves

Sine, and cosine waves are, after the impulse, the most important family of functions.

The continuous sine wave function is:

$$s(t) = A \sin(w\, t - \theta) \tag{1.41}$$

where A is the amplitude, w is the frequency, and θ is the phase. The wave signal is periodic with a period is $T = 2\pi/w$. In discrete time, the sine wave is:

$$s[n] = A \sin(w\, n - \theta) \tag{1.42}$$

where w is the frequency. Note that the discrete sine wave will not be periodic for any arbitrary value of w. A discrete signal $f[n]$ is periodic, if there exists $T \in \mathbb{N}$ such that $f[n] = f[n + mT]$ for all $m \in \mathbb{Z}$. For the discrete sine (and cosine) wave to be periodic the frequency has to be $w = 2\pi K/N$ for $K, N \in \mathbb{N}$. If K/N is an irreducible fraction, then the period of the wave will be $T = N$ samples. Although θ can have any value, here we will consider the $\theta = 0$ or $\theta = \pi/2$, which will give the sine and cosine waves respectively.

In general, to make explicit the periodicity of the wave we will use the form:

$$s_k[n] = \sin\left(\frac{2\pi}{N}\, k\, n\right) \tag{1.43}$$

the same applies for the cosine:

$$c_k[n] = \cos\left(\frac{2\pi}{N}\, k\, n\right) \tag{1.44}$$

This particular notation makes sense when considering the set of periodic signals with period N, or the set of signals with finite support signals of length N with $n \in [0, N-1]$. In such a case, $k \in [1, N/2]$ denotes the number of wave cycles that will occur within the region of support. Note that if $k = 0$ then $s[n] = 0$ and $c[n] = 1$ for all n. One can also verify that $c_{N-k} = c_k$, and $s_{N-k} = -s_k$, therefore for $k > N/2$ we find the same set of waves as the ones in the interval $s \in [1, N/2]$.

One remarkable property of sine and cosine waves is that the set of function s_k and c_k constitute what is called an orthogonal basis for all periodic discrete signals with period N, and also for all discrete signals of length N (and a similar property exists also for continuous signals). Therefore, any such signal can be written as:

$$f[n] = a_0 + \sum_{k=1}^{N/2} a_k \cos\left(\frac{2\pi}{N}\, k\, n\right) + \sum_{k=1}^{N/2} b_k \sin\left(\frac{2\pi}{N}\, k\, n\right) \tag{1.45}$$

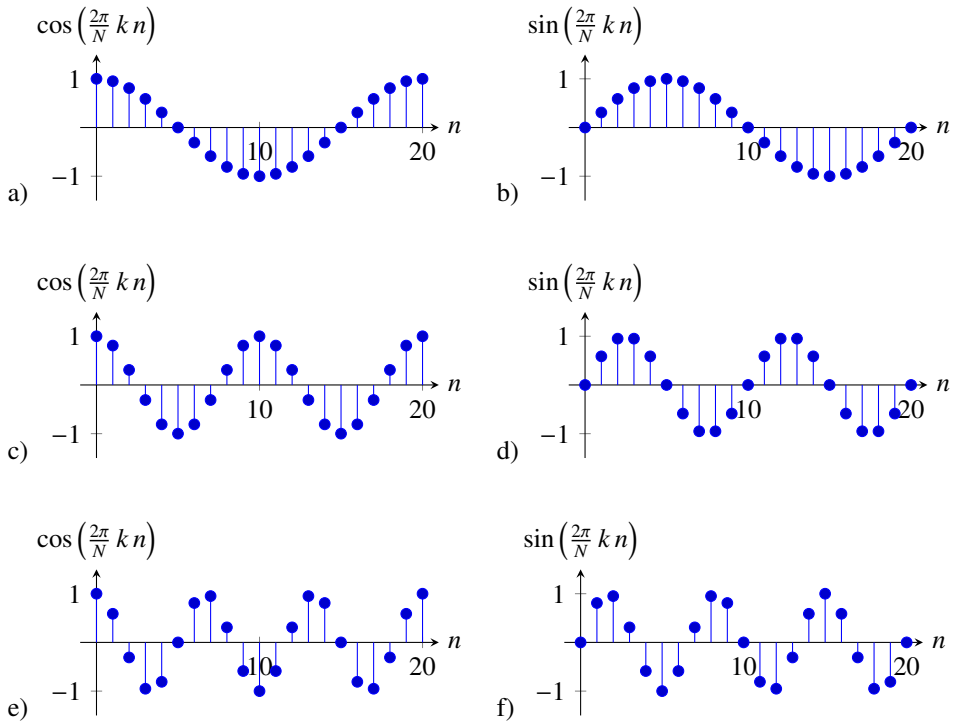


Figure 1.14
Sine and cosine waves with $A = 1$ and $N = 20$. Each row corresponds to $k = 1$, $k = 2$ and $k = 3$. Note that for $k = 3$ the waves oscillates 3 times in the interval $[0, N-1]$ but the samples in each oscillation are not identical and it is only truly periodic once every N samples. This is because $3/20$ is an irreducible fraction.

were the coefficients a_k and b_k are constants.

$$\begin{aligned} a_0 &= \frac{1}{N}\sum_{n=1}^{N-1} f[n] \\ a_k &= \frac{2}{N}\sum_{n=1}^{N-1} f[n]\cos\left(\frac{2\pi}{N}k\,n\right) \\ b_k &= \frac{2}{N}\sum_{n=1}^{N-1} f[n]\sin\left(\frac{2\pi}{N}k\,n\right) \end{aligned} \tag{1.46}$$

The set of coefficients (a_k, b_k) provide an alternative representation of the signal f to the one provided by its sample values $f[n]$. As shown in equations 1.46, the coefficients (a_k, b_k) are a linear transformation of the samples $f[n]$.

The same analysis can be extended to 2 dimensions. In 2D, the discrete sine and cosine waves are:

$$s_{u,v}[n,m] = A \sin\left(2\pi\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{1.47}$$

$$c_{u,v}[n,m] = A \cos\left(2\pi\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{1.48}$$

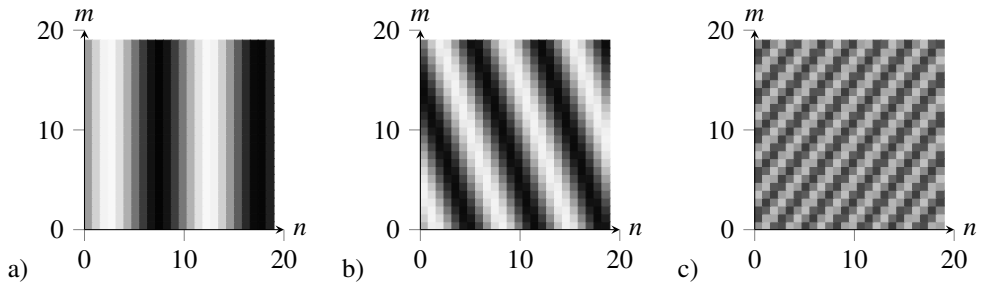


Figure 1.15
2D sine waves with $N = M = 20$. The frequency values are: a) $u = 2, v = 0$, b) $u = 3, v = 1$, c) $u = 7, v = -5$

1.5 Fourier analysis

We need a more precise language to talk about the effect of linear filters, and the different image components, than to say "sharp" and "blurry" parts of the image. The Fourier transform provides that precision. By analogy with temporal frequencies, which describe how quickly signals vary over time, a "spatial frequency" describes how quickly a signal varies over space. The Fourier transform lets us describe a signal as a sum of complex exponentials, each of a different spatial frequency.

1.5.1 Discrete Cosine Transform

As we discussed in the previous section, sine, cosine and complex exponential waves are basis for signals and images. A signal can be decomposed into a linear combination of wave functions with different frequencies. The process of going from the signal to the wave coefficients is called a transform. Going back from the coefficients into the original signal is the inverse transform.

The basis introduced in section 1.4.2 are not the only ones based in cosine and sine waves. There are many variants that can be obtained by extending the signal outside the interval $[0, N-1]$ with different kinds of symmetries.

One very important transform is the Discrete Cosine Transform (DCT), introduced in 1974 by Ahmed, Natarajan, and Rao []. There are several forms for the discrete cosine transform. Here we describe the most common one. For signals in the domain $[0, N-1]$ the DCT uses the basis:

$$\begin{aligned} x_0[n] &= \sqrt{\frac{1}{N}} \\ x_k[n] &= \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{2N} k(2n+1)\right) \quad \text{for } k \in [1, N-1] \end{aligned} \tag{1.49}$$

with $n \in [0, N-1]$. The N functions x_k are an orthonormal basis. This is:

$$\langle x_k, x_r \rangle = \sum_{n=0}^{N-1} x_k[n]\, x_r[n] = \delta[k-r] = \begin{cases} 1 & \text{if } k = r \\ 0 & \text{if } k \neq r \end{cases} \tag{1.50}$$

The discrete cosine transform (DCT) is extensively used in image processing.

The DCT of the signal $f[n]$ is:

$$F[k] = \sum_{n=0}^{N-1} f[n]\ x_k[n] \tag{1.51}$$

where $F[k]$ with $k \in [0, N-1]$ is the DCT. The inverse DCT is:

$$f[n] = \sum_{k=0}^{N-1} F[k]\, x_k[n] \tag{1.52}$$

which reconstructs the signal f from the DCT coefficients. Therefore, the DCT transform is invertible.

We can also write this in matrix form, with one basis $x_k[n]$ per row. The DCT matrix is an $N \times N$ matrix:

$$X = \begin{bmatrix} x_0[0] & x_0[1] & \dots & x_0[N-1] \\ x_1[0] & x_1[1] & \dots & x_1[N-1] \\ \vdots & \vdots & & \vdots \\ x_{N-1}[0] & x_{N-1}[1] & \dots & x_{N-1}[N-1] \end{bmatrix} \tag{1.53}$$

Note that $X X^T = I$ where I is the identity matrix. Treating the signals as vectors, we can write:

$$F = Xf \tag{1.54}$$

and

$$f = X^T F \tag{1.55}$$

It is helpful to visualize the matrix for some values of N. For instance, for $N = 4$ we can write the full matrix as:

$$X = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1.31 & 0.54 & -0.54 & -1.31 \\ 1 & -1 & -1 & 1 \\ 0.54 & -1.31 & 1.31 & -0.54 \end{bmatrix} \tag{1.56}$$

The first row is a constant vector, the second and third rows contains waves with one period with two different phases, and the last row has two periods. It is easy to check that these four row vectors are orthogonal. Figure 1.16 shows the DCT matrix, displayed as an image, for $N = 32$.

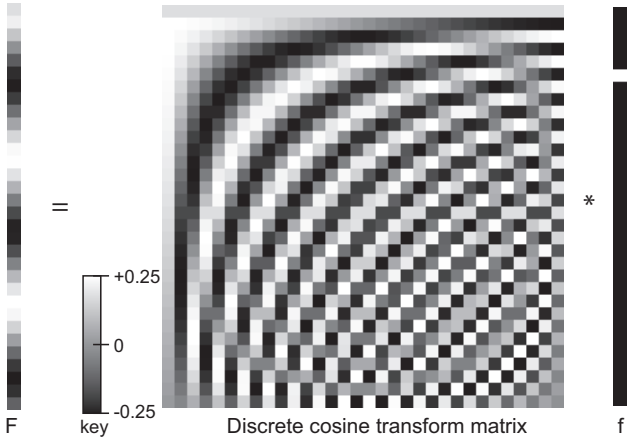


Figure 1.16
Visualization of discrete cosine transform as a matrix. The signal to be transformed forms the entries of the column vector at right. The values of the DCT matrix are indicated by the gray-scale key in the bottom left. In the vector at the right, black values indicate zero.

The DCT can be easily extended to 2D as the product of two 1D basis functions:

$$x_{u,v}[n, m] = x_u[n]\ x_v[m] \tag{1.57}$$

with x_u and x_v defined as in equation 1.49.

Although the DCT is very popular in image processing (concerned with image manipulation), it is not commonly used in computer vision (concerned with image interpretation). As a signal representation the DCT has a few issues:

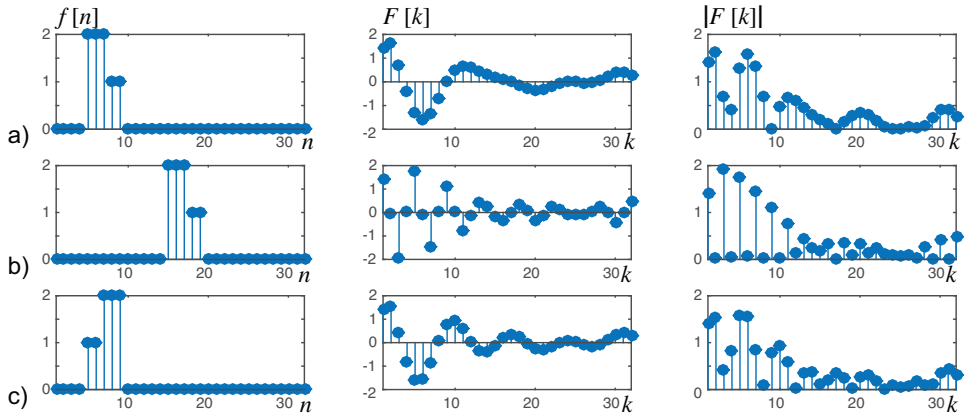


Figure 1.17
Lack of invariances in the DCT. a) a signal with $N = 32$, its DCT coefficients and the absolute value of the DCT coefficients. b) that same signal but translated. Note that both the DCT coefficients and their absolute values change. c) The same signal but with a left-right mirror. Again the DCT coefficients change. Therefore, it is difficult to use the DCT coefficients to extract properties of the signal invariant to where it is located.

- Translation invariance. Fig. 1.17 shows that the DCT coefficients change when we translate a signal, even if it is shape does not change. This is an undesirable property if our goal is to recognize the signal.
- In fact, the frequency content on a signal can not be easily analyzed. Even if you take a single sine wave and you translate it, the coefficients change in a complex way.

In the next section we will see another transform that addresses these two issues: the Fourier Transform.

1.5.2 Discrete Fourier Transform

The Fourier Transform is the most important transform and it is used in many fields.

1.5.2.1 Complex exponentials We have seen cosine and sine waves and how they can be used to transform the image into another representation. We will first describe the complex exponential, an important basis function. In continuous time, it is:

$$s(t) = A \exp(j w t) \tag{1.58}$$

with $j = \sqrt{-1}$, w is the frequency, and A is the amplitude. A can be a complex number $A = |A| exp(j\theta)$. Using Euler's formula:

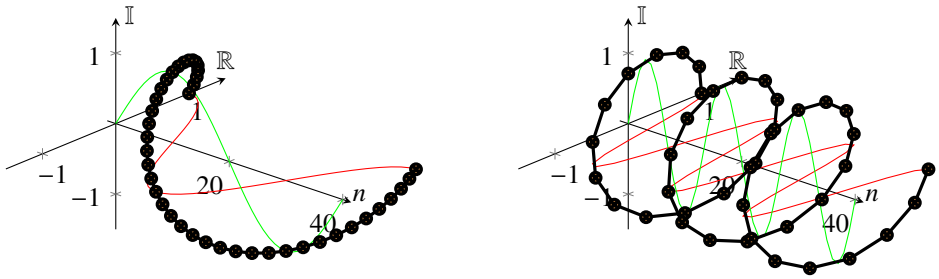
$$s(t) = A \exp(j w t) = |A| cos(wt + \theta) + j |A| sin(wt + \theta) \tag{1.59}$$

the complex exponential is a periodic function $s(t) = s(t+T)$ for $T = 2\pi/w$.

In discrete time (setting $A = 1$), we can write:

$$e_k[n] = \exp\left(j\frac{2\pi}{N}k\,n\right) = \cos\left(\frac{2\pi}{N}k\,n\right) + j\sin\left(\frac{2\pi}{N}k\,n\right) \tag{1.60}$$

Figure 1.18 shows the discrete complex exponential function. As the values are complex, the plot shows in the x axis the real component and in the y axis the imaginary component. As n goes from 0 to $N-1$ the function rotates along the complex circle of unit magnitude.



a) b)

Figure 1.18
Complex exponential wave with a) $N = 40$, $k = 1$, $A = 1$, and b) $N = 40$, $k = 3$, $A = 1$. The red and green curves show the real and imaginary waves. The yellow line is the complex exponential. The dots correspond to the discrete samples.

And in 2D, the complex exponential wave is:

$$e_{u,v}[n,m] = \exp\left(2\pi j\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{1.61}$$

where u and v are the two spatial frequencies. Note that complex exponentials in 2D are separable. This means they can be written as the product of two 1D signals:

$$e_{u,v}[n,m] = e_u[n]\,e_v[m] \tag{1.62}$$

The set of functions $e_k[n]$, with $k \in [0, N-1]$, form an orthogonal basis for discrete signals of length N. In fact,

$$\langle e_k, e_r\rangle = \sum_{n=0}^{N-1} e_k[n]\,e_r^*[n] = N\delta[k-r] = \begin{cases} N & \text{if } k = r \\ 0 & \text{if } k \neq r \end{cases} \tag{1.63}$$

Similarly, the 2D complex exponentials form a basis for discrete images of size $N \times M$:

$$\langle e_{u,v}, e_{u',v'} \rangle = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} e_{u,v}[n,m]\, e^*_{u',v'}[n,m] = MN\delta[u-u']\,\delta[v-v'] \tag{1.64}$$

Therefore, any finite length discrete signal can be decomposed as a linear combination of complex exponentials.

Complex exponentials are related to cosine and sine waves by the equalities (using Euler's formula):

$$\cos\left(2\pi\left(\frac{u\,n}{N} + \frac{vm}{M}\right)\right) = \frac{1}{2}\left(e_{u,v}[n,m] + e^*_{u,v}[n,m]\right) \tag{1.65}$$

$$\sin\left(2\pi\left(\frac{u\,n}{N} + \frac{vm}{M}\right)\right) = \frac{-j}{2}\left(e_{u,v}[n,m] - e^*_{u,v}[n,m]\right) \tag{1.66}$$

1.5.2.2 Discrete Fourier Transform and inverse Transform The Discrete Fourier Transform (DFT) transforms an image $f[m,m]$ into the complex image Fourier transform $F[u,v]$ as:

$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{1.67}$$

By applying $\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}$ to both sides of Eq. (1.72) and exploiting the orthogonality between distinct Fourier basis elements, we find the inverse Fourier transform relation:

$$f[n,m] = \frac{1}{NM} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F[u,v] \exp\left(+2\pi j\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{1.68}$$

We will call $F[u,v]$ the Fourier transform of $f[m,n]$.

As we can see from the inverse transform equation, we re-write the image, instead of as a sum of offset pixel values, as a sum of complex exponentials, each at a different frequency, called a spatial frequency for images, since they describe how quickly things vary across space. From the inverse transform formula, we see that to construct an image from a Fourier transform, capital F, we just add-in the corresponding amount of that particular complex exponential (conjugated).

As $F[u,v]$ is obtained as a sum of complex exponential with a common period of N, M samples, the function $F[u,v]$ is also periodic: $F[u+aN, v+bM] = f[u,v]$ for any $a, b \in \mathbb{Z}$. Also the result of the inverse DFT is a periodic image. Indeed you can verify from equation 1.69 that $f[n+aN, m+bM] = f[n,m]$ for any $a, b \in \mathbb{Z}$.

The DFT and its inverse in 1D are defined in the same way. We can also write the DFT in matrix form, with one basis per row. Working in 1D, as we did before, allows us visualizing the transformation matrix. Figure 1.19 shows a color visualization of the complex-valued matrix for the 1D DFT, which, when used as a multiplicand, yields the Fourier transform of 1D vectors. Many Fourier transform properties and symmetries can be observed from

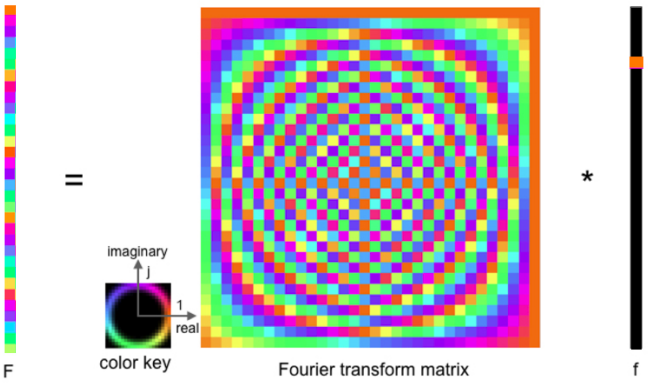


Figure 1.19
Visualization of Discrete Fourier Transform as a matrix. The signal to be transformed forms the entries of the column vector at right. The complex values of the Fourier Transform matrix are indicated by the color, with the key in the bottom left. In the vector at the right, black values indicate zero.

inspecting that matrix. Note that this matrix has also some similarities with the matrix used to compute the 1D DCT.

Using the fact that $e_{N-u,M-v} = e_{-u,-v}$, another equivalent way to write for the Fourier transform is to sum over the frequency interval $[-N/2, N/2]$ and $[-M/2, M/2]$. This is specially useful for the inverse that can be written as:

$$f[n,m] = \frac{1}{NM} \sum_{u=-N/2}^{N/2} \sum_{v=-M/2}^{M/2} F[u,v] \exp\left(+2\pi j \left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{1.69}$$

This formulation allows us to arrange the coefficients in the complex plane so that the zero frequency, or "DC", coefficient is at the center. Slow, large variations correspond to complex exponentials of frequencies near the origin. If the amplitudes of the complex conjugate exponentials are the same, then their sum will represent a cosine wave; if their amplitudes are opposite, it will be a sine wave. Frequencies further away from the origin represent faster variation with movement across space.

The DFT often provides a better representation than the DCT. Note that, as both are invertible transforms, it is possible to compute the DCT coefficients from the DFT coefficients (and vice-versa), but the DFT makes explicit signal information (such as the frequency content and translation invariance of the coefficient magnitudes) that is useful for image interpretation.

We will see later some properties of the Fourier transform. But one very important property is that the decomposition of a signal into a sum of complex exponentials is unique: there is a unique linear combination of the exponentials that will result in a signal.

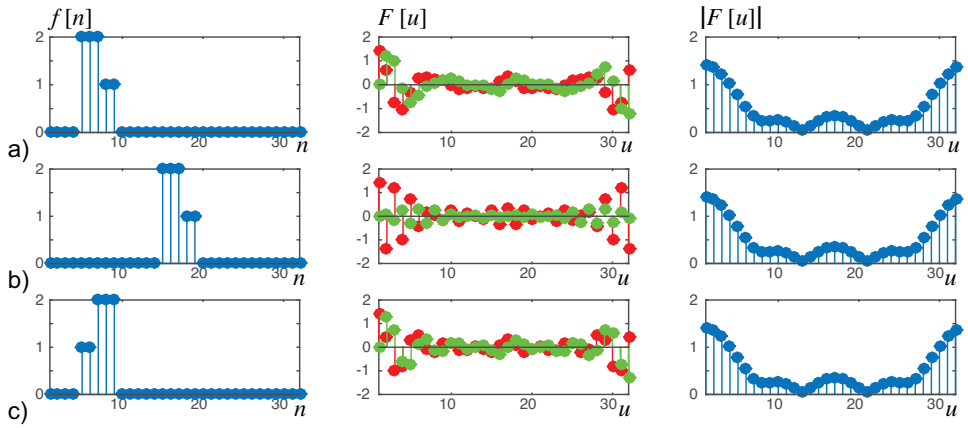


Figure 1.20
Invariances in the DFT (compare with fig. 1.17). a) a signal with $N = 32$, its DFT coefficients (real and imaginary parts shown in red and green) and the magnitude of the DFT coefficients. b) that same signal but translated. Note that both the magnitude of the DFT coefficients has not changed. c) The same signal but with a flipped left-right. Again the DFT coefficient magnitude has not change. Therefore, the DFT coefficients provide a translation invariant signal representation.

1.5.2.3 Discrete Fourier Transform of real images Let's now look at the DFT of a real picture. In this case we will not be able to write the analytic form of the result, but there are a number of properties that will hold and that will help us to interpret the result.

Figure 1.21 shows the Fourier Transform of a 64×64 resolution image of a cube. As the DFT results in a complex representation, there are two possible ways of writing the result. Using the real and imaginary components:

$$F[u,v] = Re\{F[u,v]\} + j\, Imag\{F[u,v]\} \tag{1.70}$$

where *Re* and *Imag* denote the real and imaginary part of each Fourier coefficient. Or using a polar decomposition:

$$F[u,v] = A[u,v]\, \exp(j\,\theta[u,v]) \tag{1.71}$$

where $A[u,v] \in \mathbb{R}^+$ is the amplitude and $\theta[u,v] \in [-\pi, \pi]$ is the phase. Figure 1.21 shows both decompositions of the Fourier transform.

Although the DFT is a complex transform (the basis functions are vectors of complex numbers), we will use it to represent images of real numbers. This means that when we use the inverse Fourier transform to recover the real image from its complex Fourier transform, the imaginary coefficients of the Fourier Transform will have to cancel out. Indeed, really we're describing the image as a sum of sines and cosines, which we'll create from the complex exponentials by taking sums and differences of them, at the same amplitude. So

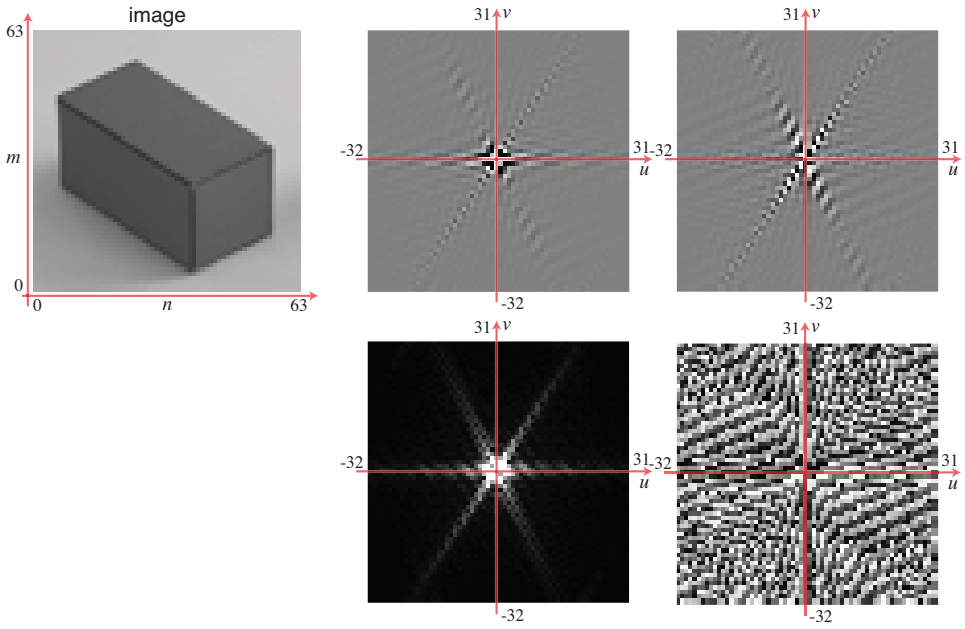


Figure 1.21
DFT of an image.

to generate a real valued image, the Fourier transform will always have real component that is even, and an imaginary component that is odd. We can show this from the definition of the Fourier transform:

$$F^*[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m]\exp\left(2\pi j\left(\frac{u\,n}{N}+\frac{v\,m}{M}\right)\right) = F[-u,-v] = F[N-u,M-v] \tag{1.72}$$

So, the Fourier transform of a real signal has coefficients that come in pairs, with $F[u,v]$ being the complex conjugate of $F[-u,-v]$. From this we can show that the real and imaginary parts of the Fourier transform will have the symmetries:

$$Re\{F[u,v]\} = Re\{F[-u,-v]\} \tag{1.73}$$

$$Imag\{F[u,v]\} = -Imag\{F[-u,-v]\} \tag{1.74}$$

If the image was composed only of imaginary numbers, then $F^*[u,v] = -F[-u,-v]$. It is easy to show that if the image is symmetric, $f[n,m] = f[N-n,M-m]$, then the DFT is real (i.e., phase of the DFT is 0).

Upon first learning about Fourier transforms, it may be a surprise to learn that one can synthesize any image as a sum of complex exponentials (sines and cosines). To help gain insight into how that works, it is informative to show examples of partial sums of complex exponentials. Figure 1.22 shows partial sums of the Fourier components of an image. In each partial sum of N components, we use the largest N components of the Fourier transform. Using the fact that the Fourier basis functions are orthonormal, it is straightforward to show that this is the best least squares reconstruction possible from each given number of Fourier basis components. This first image shows what is reconstructed from the largest Fourier component which turns out to be $F[0,0]$. This component encodes the DC value of the image, therefore the resulting image is just a constant. The next two components correspond to two complex conjugates of a very slow varying wave. And so on. As more components get added, the figure slowly emerges. In this example, the first 127 coefficients are sufficient for recognizing this 64x64 resolution image.

1.5.2.4 Useful transforms It's useful to become adept at computing and manipulating simple Fourier transforms. Figure 1.23 shows a list of useful Fourier transform pairs (temporarily showing figures from Bracewell and Szeliski's books), and these are useful to study and become familiar with.

For some simple cases, we can compute the analytic form of the Fourier transform.

Fourier transform of the Delta function $\delta[n,m]$:

$$F[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} \delta[n,m]\exp\left(-2\pi j\left(\frac{u\,n}{N}+\frac{v\,m}{M}\right)\right) = 1 \tag{1.75}$$

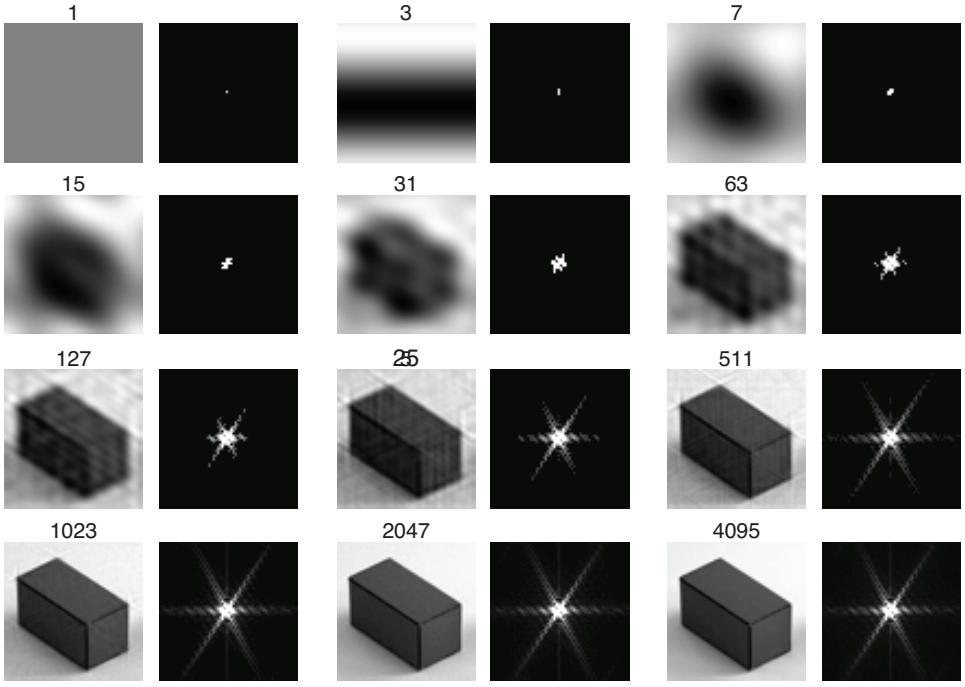
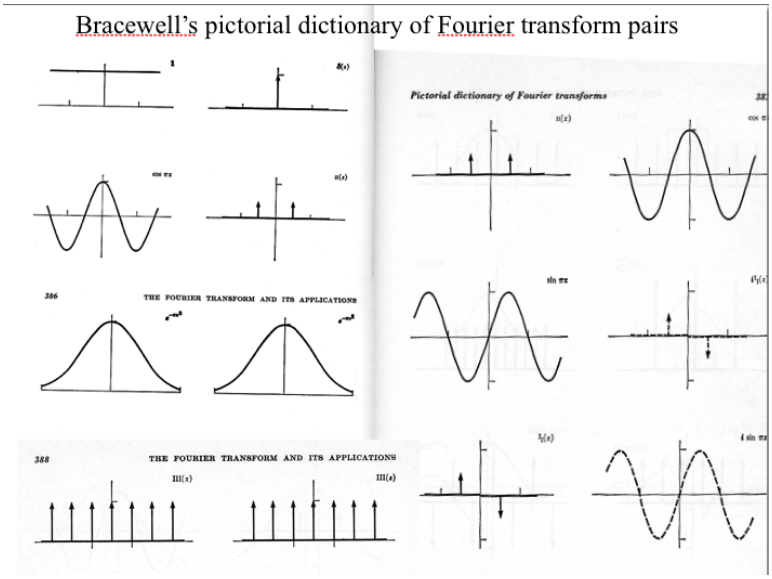


Figure 1.22
Reconstructing an image from the N Fourier coefficients of the largest amplitude. The left frame shows the location, in the Fourier domain, of the N Fourier coefficients which, when inverted, give the image at the right. Using only 1025 coefficients, the image is seen clearly.



(a)

Name	Signal		Transform
impulse	$\delta(x)$	$\Leftrightarrow$	1
shifted impulse	$\delta(x-u)$	$\Leftrightarrow$	$e^{-j\omega u}$
box filter	$\text{box}(x/a)$	$\Leftrightarrow$	$a\text{sinc}(a\omega)$
tent	$\text{tent}(x/a)$	$\Leftrightarrow$	$a\text{sinc}^2(a\omega)$
Gaussian	$G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$
Laplacian of Gaussian	$(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2})G(x;\sigma)$	$\Leftrightarrow$	$-\frac{\sqrt{2\pi}}{\sigma}\omega^2 G(\omega;\sigma^{-1})$
Gabor	$\cos(\omega_0 x)G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega\pm\omega_0;\sigma^{-1})$
unsharp mask	$(1+\gamma)\delta(x)$ $-\gamma G(x;\sigma)$	$\Leftrightarrow$	$(1+\gamma)-$ $\frac{\sqrt{2\pi}\gamma}{\sigma}G(\omega;\sigma^{-1})$
windowed sinc	$\text{rcos}(x/(aW))$ $\text{sinc}(x/a)$	$\Leftrightarrow$	(see Figure 3.29)

Table 3.2 Some useful (continuous) Fourier transform pairs: The dashed line in the Fourier

Szeliski, Computer Vision, 2010

(b)

Figure 1.23
(a) and (b): A collection of useful Fourier transform pairs, from [**?**] and [**?**].

the Fourier transform of the delta signal is a constant. If we think in terms of the inverse Fourier transform, this means that if we sum all the complex exponentials with a coefficient of 1, then all the values will cancel but the one at the origin which results in a delta function:

$$\delta[n,m] = \frac{1}{NM}\sum_{u=-N/2}^{N/2}\sum_{v=-M/2}^{M/2}\exp\left(2\pi j\left(\frac{u\,n}{N}+\frac{v\,m}{M}\right)\right) \tag{1.76}$$

The Fourier transform of the cosine wave, $\cos\left(2\pi\left(\frac{u_0\,n}{N}+\frac{v_0\,m}{M}\right)\right)$, is:

$$\begin{aligned} F[u,v] &= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\cos\left(2\pi\left(\frac{u_0\,n}{N}+\frac{v_0\,m}{M}\right)\right)\exp\left(-2\pi j\left(\frac{u\,n}{N}+\frac{v\,m}{M}\right)\right) = & (1.77)\\ &= \frac{1}{2}\left(\delta[u-u_0,v-v_0]+\delta[u+u_0,v+v_0]\right) & (1.78)\end{aligned}$$

this can be easily proven using Euler's equation 1.65 and the orthogonality between complex exponentials. And for the sine wave, $\sin\left(2\pi\left(\frac{u_0\,n}{N}+\frac{v_0 m}{M}\right)\right)$, we have a very similar relationship:

$$F[u,v] = \frac{1}{2j}\left(\delta[u-u_0,v-v_0]-\delta[u+u_0,v+v_0]\right) \tag{1.79}$$

Figure 1.24 shows the DFT of several waves with different frequencies and orientations.

Figure 1.25 shows the 2-d Fourier transforms of some simple signals. The depicted signals all happen to be symmetric about the spatial origin. From the Fourier transform equation, one can show that real and even input signals transform to real and even outputs. So for the examples of Fig. 1.25, we only show the magnitude of the Fourier transform, which in this case is the absolute value of the real component of the transform, and the imaginary component happens to be zero for the signals we'll examine. Also, all these images but the last one are separable (they can be written as the product of two 1D signals). Therefore, their DFT is also the product of 1D DFTs from figure 1.23.

1.5.2.5 Discrete Fourier transform properties For now, when we talk about images or signals we will assume they are periodic signals with periods N and M in each dimension.

Linearity The Fourier transform and its inverse are linear transformations:

$$DFT\{\alpha f[n,m]+\beta g[n,m]\} = \alpha F[u,v]+\beta G[u,v] \tag{1.80}$$

where α and β are complex constants.

Separability An image is separable if it can be written as the product of two 1D signals, $f[n,m] = f_1[n]\,f_2[m]$. If the image is separable, then its Fourier transform is separable: $F[u,v] = F_1[u]\,F_2[v]$

Shift: translation in space If we displace a signal in the spatial domain, it results in multiplying its Fourier transform by a complex exponential.

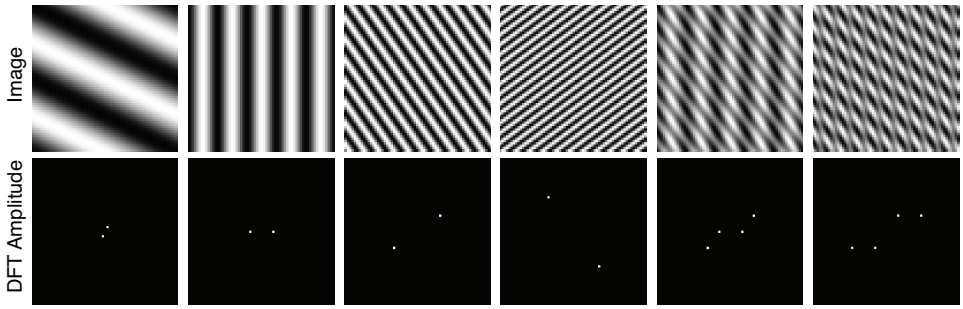


Figure 1.24
Some two-dimensional Fourier transform pairs. Images are 64×64 pixels. The waves are *cos* with frequencies $(1,2)$, $(5,0)$, $(10,7)$, $(11,-15)$. The last two examples show the sum of two waves and the product.

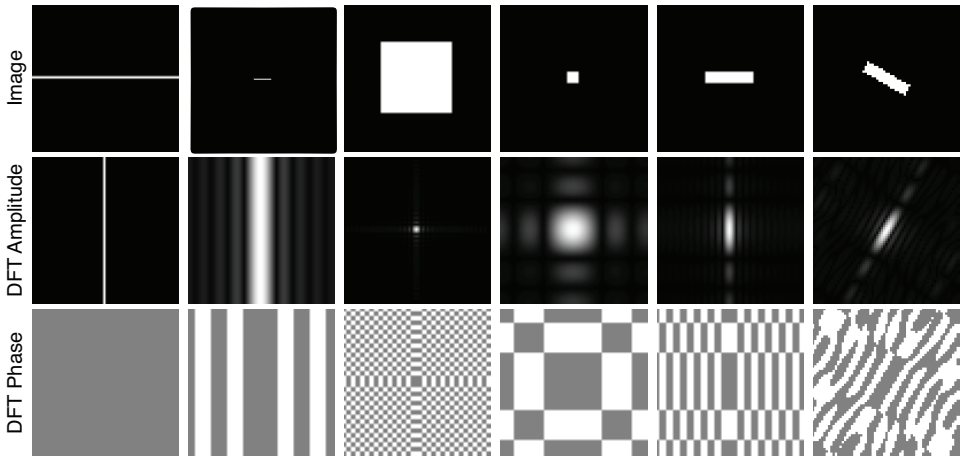


Figure 1.25
Some two-dimensional Fourier transform pairs. Note the trends visible in the collection of transform pairs: As the support of the image in one domain gets larger, the magnitude in the other domain becomes more localized. A line transforms to a line oriented perpendicularly to the first. Images are 64×64 pixels.

To show this, consider an image $f[n,m]$, with Fourier Transform $F[u,v]$ and period N,M. When displacing the image by n_0, m_0 pixels, we get $f[n-n_0, m-m_0]$ and its Fourier Transform is:

$$\begin{aligned} DFT\{f[n-n_0,m-m_0]\} = \\ &= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n-n_0,m-m_0]\exp\left(-2\pi j\left(\frac{u\,n}{N}+\frac{v\,m}{M}\right)\right) = \\ &= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m]\exp\left(-2\pi j\left(\frac{u\,(n+n_0)}{N}+\frac{v\,(m+m_0)}{M}\right)\right) = \\ &= F[u,v]\exp\left(-2\pi j\left(\frac{u\,n_0}{N}+\frac{v\,m_0}{M}\right)\right) \end{aligned} \tag{1.81}$$

Note that as the signal f and the complex exponentials have the period N,M, we can change the sum indices over any range of size $N \times M$ samples.

Note that in practice, if we have an image and we apply a translation there will be some boundary artifacts. So, in general, this property is only true if we apply a circular translation. Otherwise, it will be only an approximation. Fig. 1.26 shows two images that correspond to a translation with $n_0 = 16$ and $m_0 = -4$. Note that at the image boundaries, new pixels appear in (c) not visible in (a). As this is not a pure circular translation, the result from eq. 1.81 will not apply exactly. To verify eq. 1.81 let's look at the real part of the DFT of each image shown in fig. 1.26.b and d. If eq. 1.81 holds true, then the real part of the ratio between the DFTs of the two translated images should be $\cos\left(-2\pi j\left(\frac{u\,n_0}{N}+\frac{v\,m_0}{M}\right)\right)$ with $N = M = 128$ and $[n_0, m_0] = [16, -4]$. Fig. 1.26.f shows that the real part of the ratio is indeed very close to a cosine, despite of the boundary pixels which are responsible of the noise (the same is true for the imaginary part). In fact, fig. 1.26.e shows the inverse DFT of the ratio between DFTs, considering both real and imaginary components, which is very close to an impulse at $[16, -4]$.

Locating the maximum on Fig. 1.26.f can be used to estimate the displacement between two images when the translation corresponds to a global translation. However, this method is not very robust and it is rarely used in practice.

Modulation: Translation in frequency If we multiply an image with a complex exponential, its Fourier Transform is translated, a property related to the previous one:

$$DFT\left\{f[n,m]\exp\left(-2\pi j\left(\frac{u_0\,n}{N}+\frac{v_0\,m}{M}\right)\right)\right\} = F[u-u_0, v-v_0] \tag{1.82}$$

Note that now the image is not real anymore, and for this reason its Fourier Transform does not has symmetries around $u, v = 0$.

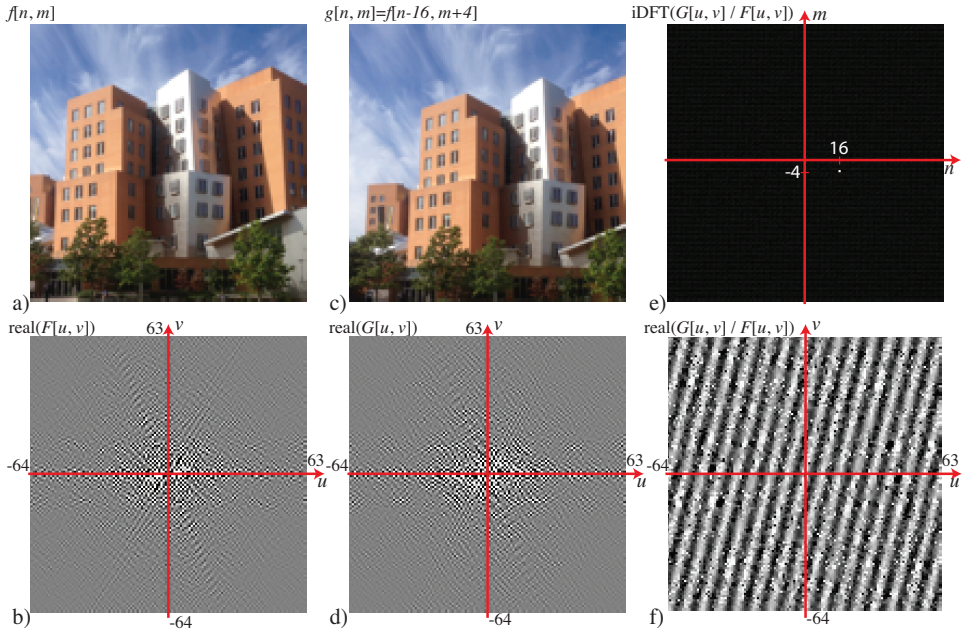


Figure 1.26
Translation in space. Image (c) corresponds to image (a) after a translation of 16 pixels to the right and 4 pixels down. Images (b) and (d) show the real parts of their corresponding DFTs (with $N = 128$). The image (f) shows the real part of the ratio between the two DFTs, and (e) is the inverse transform of the ratio between DFTs. The inverse is very close to an impulse located at the coordinates of the displacement vector between the two images.

A related relationship is:

$$DFT\left\{f[n,m]\cos\left(2\pi j\left(\frac{u_0\,n}{N}+\frac{v_0\,m}{M}\right)\right)\right\}=F[u-u_0,v-v_0]+F[u+u_0,v+v_0] \quad (1.83)$$

Multiplying a signal by a wave is called signal modulation and it is one of the basic operations in communications. It is also an important property in image analysis and we will see its use later.

Note that a shift and a modulation are equivalent operations in different domains. A shift in space is a modulation in the frequency domain and that a shift in frequency is a modulation in the spatial domain.

Parseval's theorem As the DFT is a change of basis, the dot product between two signals and the norm of a vector is preserved (up to a constant factor) after the basis change. This is stated by Parseval's theorem:

$$\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m]\,g^*[n,m]=\frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} F[u,v]\,G^*[u,v] \quad (1.84)$$

and, in particular, if $f=g$ this reduces to the Plancherel theorem:

$$\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} \|f[n,m]\|^2=\frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \|F[u,v]\|^2 \quad (1.85)$$

This relationship is important because it tells us that the energy of a signal can also be computed as a sum of the squared magnitude of the values of its Fourier transform.

Convolution The Fourier transform lets us characterize images by their spatial frequency content. It's also the natural domain in which to analyze space invariant linear processes, because the Fourier bases are the eigenfunctions of all space invariant linear operators. In other words, if you start with a complex exponential, and apply any linear, space invariant operator to it, you always come out with a complex exponential of that same frequency, but, in general, with some different amplitude and phase.

Another way to state that property is through the Fourier convolution theorem, given below. Consider a function f that is the convolution of two functions, g and h:

$$f=g\circ h \quad (1.86)$$

If we take the Fourier transform of both sides, and use the definition of the Fourier transform, we obtain

$$\begin{aligned} F[u,v] &= DFT\{g\circ h\} \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g[m-k,n-l]\,h[k,l]\exp\left(-2\pi j\left(\frac{mu}{M}+\frac{nv}{N}\right)\right) \end{aligned} \quad (1.87)$$

Changing the dummy variables in the sums (introducing $m' = m - k$ and $n' = n - l$), we have

$$F[u,v] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} h[k,l] \sum_{m'=-k}^{M-k-1}\sum_{n'=-l}^{N-l-1} g[m',n'] \exp\left(-2\pi j\left(\frac{(m'+k)u}{M} + \frac{(n'+l)v}{N}\right)\right) \quad (1.88)$$

Recognizing that the last two summations give the DFT of $g[n,m]$, using circular boundary conditions, gives

$$F[u,v] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} G[u,v] \exp\left(-2\pi j\left(\frac{ku}{M} + \frac{lv}{N}\right)\right) h[k,l] \quad (1.89)$$

Performing the DFT indicated by the second two summations gives the desired result,

$$F[u,v] = G[u,v]\, H[u,v] \quad (1.90)$$

Thus, the operation of a convolution, in the Fourier domain, is just a multiplication of the Fourier transform of each term in the Fourier domain. This property lets us examine the operation of a filter on any image by examining how it modulates the Fourier coefficients of any image.

Dual convolution The Fourier transform of the product of two images

$$f[n,m] = g[n,m]\, h[n,m] \quad (1.91)$$

is the convolution of their DFTs:

$$F[u,v] = \frac{1}{NM} G[u,v] \circ H[u,v] \quad (1.92)$$

1.5.3 Continuous Fourier Transform

For signals defined on the continuous domain, the Fourier Transform is defined as:

$$F(w_x, w_y) = \int\int_{-\infty}^{\infty} f(x,y) \exp\left(-j\left(w_x x + w_y y\right)\right) dx\, dy \quad (1.93)$$

and its inverse is:

$$f(x,y) = \int\int_{-\infty}^{\infty} F(w_x, w_y) \exp\left(j\left(w_x x + w_y y\right)\right) dw_x\, dw_y \quad (1.94)$$

Most of the equations that we have seen for the DFT also apply to the continuous domain, replacing sums with integrals.

The convolution between two continuous signals is written as:

$$f(x,y) = h \circ g = \int\int_{-\infty}^{\infty} h(x-x', y-y') g(x', y')\, dx'\, dy' \quad (1.95)$$

Although, in practice, images and filters will be discrete signals, many times it is convenient to think of them as continuous signals.

1.5.4 Fourier analysis as an image representation

The Fourier Transform has been extensively used as an image representation. In this section we will discuss the information about the picture that is made explicit by this representation.

1.5.4.1 Amplitude and Phase As we discussed before, the Fourier transform of an image can be written in polar form:

$$F[u,v] = A[u,v] \exp(j\theta[u,v]) \tag{1.96}$$

where

$$A[u,v] = |F[u,v]| \tag{1.97}$$

and

$$\theta[u,v] = \angle F[u,v] \tag{1.98}$$

If we think in terms of the inverse of the Fourier transform, $A[u,v]$ gives the strength of the weight for each complex exponential and the phase $\theta[u,v]$ translates the complex exponential. The phase carries the information of where the image contours are, by specifying how the phases of the sinusoids must line up in order to create the observed contours and edges. In fact, as shown in section **??**, translating the image in space only modifies the phase of its Fourier transform. In short, one can think that location information goes into the phase while intensity scaling goes into the magnitude.

One might ask which is more important in determining the appearance of the image, the magnitude of the Fourier transform, or its phase. Figure 1.27 shows the result of a classical experiment that consists in computing the Fourier transform of two images and building two new images by swapping their phases []. The first output image is the inverse Fourier transform of the amplitude of the first input image and the phase of the DFT of the second input image. The second output image contains the other two terms. The figure shows that the appearance of the resulting images is mostly dominated by the phase of the image they come from. The image built with the phase of the stop sign looks like the stop sign even if the amplitude comes from a different image. Figure 1.27 shows the result in color by doing the same operation over each color channel (R, G and B) independently. The phase signal determines where the edges and colors are located in the resulting image. The final colors are altered as the amplitudes have changed.

As we will discuss in chapter **??**, one remarkable property of real images is that the magnitude of the DFT of natural images are quite similar one to another and can be approximated by $A[u,v] = a/(u^2+v^2)^b$ with a and b being two constants.

However, this does not mean that all the information of the image is contained inside the phase only. The amplitude contains very useful information as shown in fig. 1.28. To get an intuition of the information available on the amplitude and phase let's do the following experiment: let's take an image, compute the Fourier transform and create two images

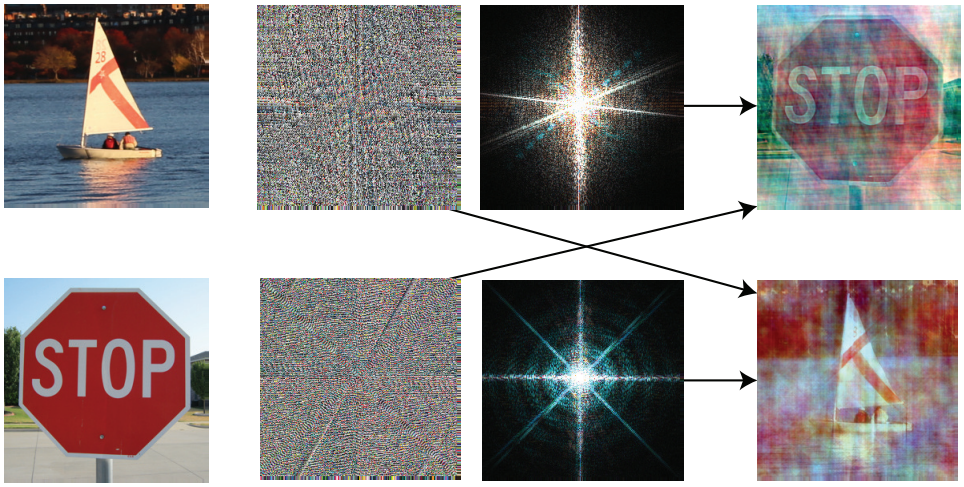


Figure 1.27
Swapping the amplitude and the phase of the Fourier Transform of two images. Each color channel is processed in the same way.

by applying the inverse Fourier transform when removing one of the components while keeping the other original component. For the amplitude image, we will randomize the phase. For the phase image, we will replace the amplitude by a non-informative $A[u, v] = 1/(u^2 + v^2)^{1/2}$ for all images. This amplitude is better than a random amplitude because a random amplitude produces a very noisy image hiding the information available, while this generic form for the amplitude will produce a smoother image revealing its structure while still removing any information available on the original amplitude. Fig. 1.28 shows different types of images and how the DFT amplitude and phase contribute to define the image content. The top image is inline with the observation from fig. 1.27 where phase seems to be carrying most of the image information. However, the rest of the images do not show the same pattern.

The amplitude is great for capturing images that contain strong periodic patterns. In such cases, the amplitude can be better than the phase. This observation has been the basis for many image descriptors [**? ?**]. The amplitude is somewhat invariant to location (although it is not invariant to the relative location between different elements in the scene). However the phase is a complex signal that does not seem to make explicit any information about the image.

Exercise: reproduce the figure 1.28 but in color. For this to work, it is better to do PCA in color space first to rotate the color space to three channels that are decorrelated, then build each decorrelated channel independently and then merge undoing the rotation to get

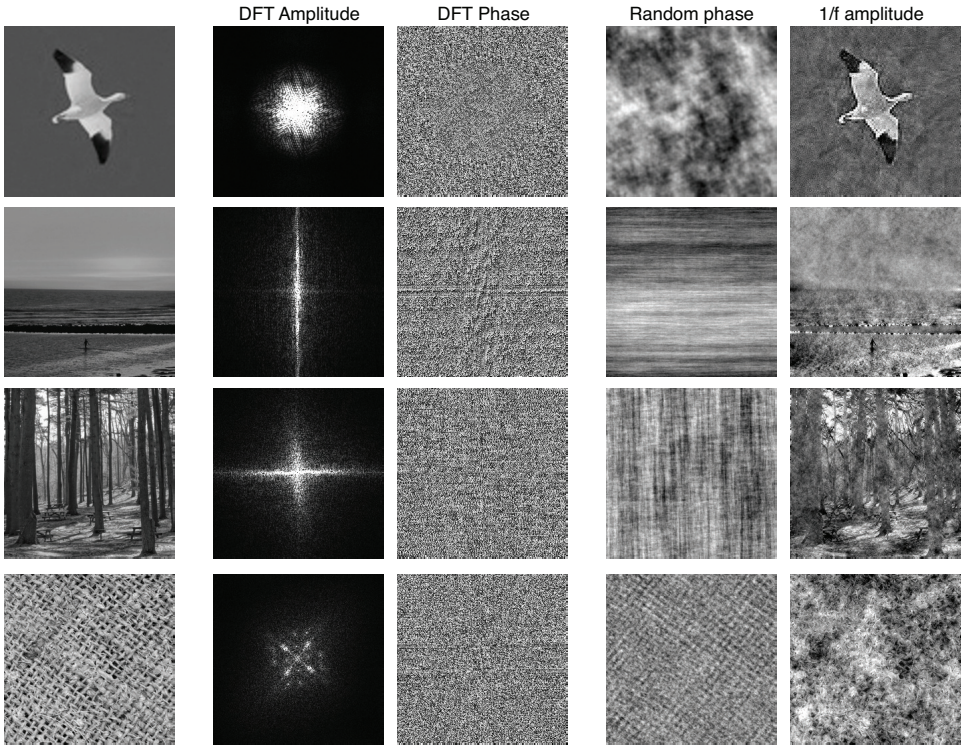


Figure 1.28
The relative importance of phase and amplitude depends on the image. Each row shows one image, its Fourier transform (amplitude and phase), and the resulting images obtained by applying the inverse Fourier transform to a signal with the original amplitude and randomized phase, and a signal with the original phase and a generic fixed $1/f$ amplitude. Note that for the first image, the phase seems to be the most important component. However, as we move down, the relative importance between the two components changes. And for the bottom image (showing a pseudo-periodic threat texture) the amplitude is the most important component.

an RGB image. In the color chapter we will see better examples of why this rotation is important.

1.5.4.2 Orientations and scales (needs to be written)

Another way in which the Fourier transform is useful is because it makes explicit which components contribute to different scales and different orientations.

Show some examples to illustrate how to read orientation and scale.

FIGURE: rocks of different sizes. textures of different orientations.

1.5.4.3 Images in the Fourier domain (needs to be written)

- show pictures in which basic frequencies are visible: waves in the water, images with strong periodic patterns and show their fourier transforms

- table of some basic images and their 2D fourier transforms (e.g, a wave, a rectangle, a circle, an oriented line, a segment, a dot, ...)

- discuss the issue about visualizing the power spectrum of natural images to avoid DC component. Also, use a window to avoid the vertical and horizontal lines in the power spectrum due to the boundary artifacts.

- game A B C, 1 2 3: maybe make it add a few more examples to make it more challenging and interesting.

IDEA: can we extract the pattern that gets repeated?

Based on this example, and the Fourier transform pairs of Fig. 1.25, take the following quiz: match these Fourier transform magnitudes with the corresponding images in Fig. 1.29

1.5.4.4 Filters in the Fourier domain Some image patterns are easily visible in the Fourier domain. For instance, strong image contrasts produce oriented lines in the Fourier domain. Periodic patterns are also clearly visible in the Fourier domain. A periodic pattern in the image domain produces picks in the Fourier domain. The location of the picks will be related to the period and orientation or the repetitions.

Fig. 1.30.a shows a picture of the main MIT building. The columns produce a quasi periodic pattern. Fig. 1.30.b shows the magnitude of the DFT of the MIT picture. One can see picks in the horizontal frequency axis, those picks are due to the columns. To check this we can verify first that the location of the picks is related to the separation of the columns. The image in Fig. 1.30.a has a size of 256×256 pixels, and the columns are are repeated each 14 pixels. Therefore, the DFT, with $N = 256$, will have picks at the horizontal frequencies: $256/14 = 18.2$, which is indeed what we observe in Fig. 1.30.b. As the repeated pattern is not a pure sinusoidal function, there will be picks at all the harmonic frequencies $k\frac{256}{14}$, where k is an integer. Note also that the picks seem to produce vertical bands with decreasing amplitude with increasing vertical frequency v. These bands are to the fact that the columns only occupy a small vertical segment of the image. Also, as the columns only exist in a portion of the horizontal region of the image, the picks have also some horizontal width.

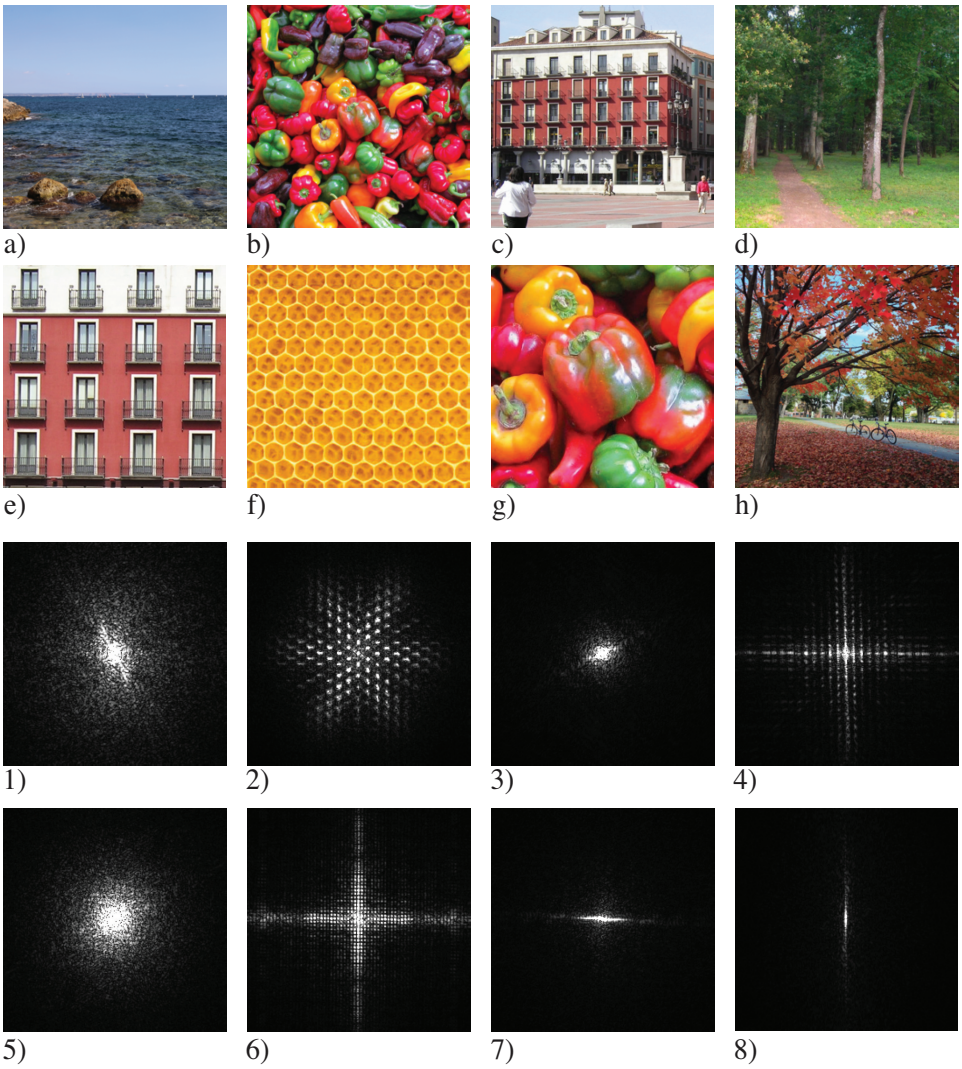


Figure 1.29
The Fourier transform matching game: Match each image (a-h) with its corresponding Fourier transform magnitude (1-8). The correct answer is: 1-h, 2-f, 3-g, 4-c, 5-b, 6-e, 7-d, 8-a.

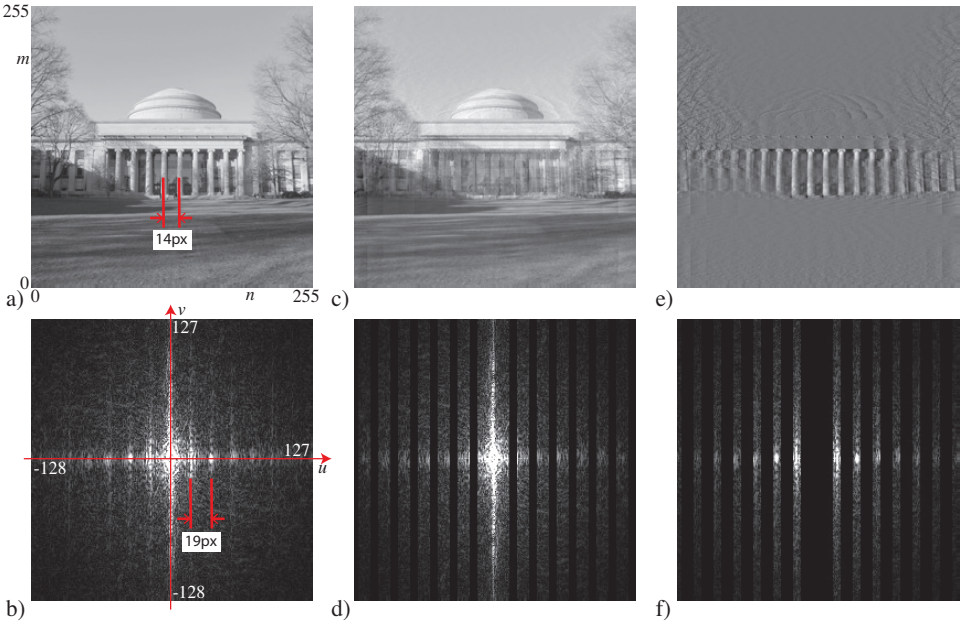


Figure 1.30
Simple filtering in the Fourier domain. (a) The repeated columns of the building of the MIT dome generate harmonics along a horizontal line in the Fourier domain. (b) By zeroing out those Fourier components, the columns of the building are substantially removed.

We can now also check the effect of suppressing those frequencies by zeroing the magnitude of the DFT around each pick (here we zero 7 pixels in the horizontal dimension and all the pixels along the vertical dimension) as shown Fig. 1.30.d. Fig. 1.30.c shows the resulting image where the columns are almost gone while the rest of the image is little affected. Fig. 1.30.e shows the complementary image (in fact $a = c + e$) and its DFT Fig. 1.30.f.

1.6 Sampling

Sampling is the process of transforming a continuous signal into a discrete one. In nature, most of the signals we measure (sound, light, ...) are defined over continuous domains (time, space, ...). In order to process them with computers we need to transform the continuous domain into a discrete one. This process is called sampling.

We need to study the following questions: what are the possible sampling patterns to discretize a signal? how can we characterize the lost of information? and how do we reduce artifacts?

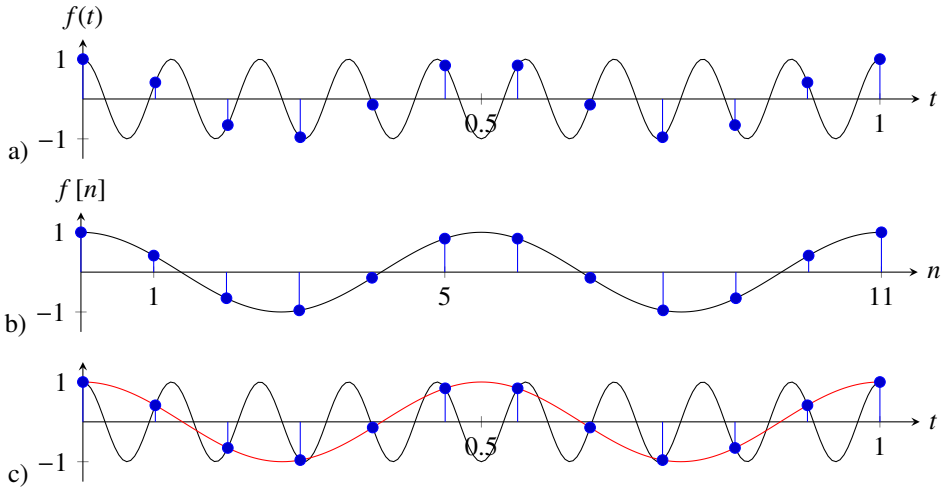


Figure 1.31
a) Continuous signal and its samples. b) Discrete signal and the reconstructed continuous signal by interpolation. c) Superposition of continuous signal (a) and its reconstructed approximation from the discrete samples from (b).

Let's consider a 1D continuous time signal $f(t)$ and its sampled version $f[n] = f(nT_s)$, were T_s is the sampling period. Intuitively, it is clear that in this sampling process some information will get lost. If no information was lost, then we should be able to recover the continuous signal $f(t)$ from its sampled version $f[n]$ by doing some kind of interpolation. One could simply decrease T_s, which will results in a more accurate approximation of the continuous signal $f(t)$ at the expense of the amount of memory needed to store $f[n]$. Decreasing T_s will also result in an increase of the computational cost of processing the signal $f[n]$. Therefore, it is interesting choosing the appropriate T_s. Understanding the sampling process and how to reconstruct the continuous signal is important as it will allow us to find the optimal sampling parameters.

1.6.1 Sampling theorem

Let's first look at one example to get a sense of the type of issues that might arise when discretizing a signal. Figure 1.31.a shows one continuous signal with the form $f(t) = \cos(wt)$ with $w = 18\pi$. The period of this signal is $T = 1/9$ (there are 9 periods in the interval $t \in [0, 1]$). We now build a discrete signal $f[n] = f(nT_s)$ with $T_s = 1/11$ (there are 11 samples in the same interval $t \in [0, 1]$). This could seem enough because there are more samples than periods.

Figure 1.31.b shows $f[n]$. If we now want to reconstruct the original continuous signal from its samples $f[n]$ there are many possibilities as the samples do not constraint what happens between samples. Therefore we will need to make some assumptions about the continuous signal. In the absence of any other prior information, we will assume that the most likely signal is the slowest and smoothest signal (we will make this assumption more precise later). Figure 1.31.c shows the superposition of the original signal and the reconstructed one. Both signals perfectly pass through the same samples. Clearly the samples seem to correspond to a cosine function with a lower frequency (in this example $T = 1/2$) than the input (which had $T = 1/9$).

It is important to mention that there is nothing special on how the parameters have been chosen for this example. Many different parameter choices would have yielded the same qualitative behavior. This confusion of frequencies is called *aliasing*. We will show that for the reconstruction to match the input we need $T_s < 1/(2T)$. The *sampling theorem* (also known as Nyquist theorem) states that for a signal to be perfectly reconstructed from it samples, the sampling frequency $f_s = 1/T_s$ has to be $f_s > 2f_{max}$ when f_{max} is the maximum frequency present in the input signal. You can check that our previous example did not satisfied the Nyquist condition.

One way of characterizing the sampling process is achieved by analyzing the relationship between the Fourier transform of the continuous and discrete signals. There are many ways of finding the relationship between the two Fourier transforms. Here we will describe the most common one.

Let's start writing a model of the sampling process by defining a special signal:

$$\hat{f}(t) = f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} f(nT_s)\delta(t - nT_s) = f(t)\delta_{T_s}(t) \tag{1.99}$$

where $\hat{f}(t)$ is a very special function that contains the same amount of information as the discrete signal $f[n]$ but that is defined over the continuous domain t. Note that $\hat{f}(t)$ is the product of two continuous signals. The first term is the continuous signal $f(t)$, the second term is a function composed of impulses placed at regular time instants $\delta(t - nT_s)$. The use of impulses is interesting because they are infinitely narrow in time, so the product of an impulse with a function is equivalent to taking just one sample of that function. Remember from the definition of $\delta(t)$ that $f(t)\delta(t - nT_s) = f(nT_s)\delta(t - nT_s)$. We define the impulse train (also called Dirac comb), $\delta_{T_s}(t)$, as the signal:

$$\delta_{T_s}(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \tag{1.100}$$

Although we will never directly work with the signal $\hat{f}(t)$, it is a convenient construction to understand how information is transformed during the sampling process. To see the

interest of this construction, let's compute its Fourier transform. The continuous Fourier transform of $\hat{f}$ can be written as the convolution of the Fourier transforms of $f(t)$ and $\delta_{T_s}(t)$.

The Fourier transform of a Delta comb is:

$$\begin{aligned}
\Delta_{T_s}(w) &= \int_{-\infty}^{\infty} \delta_{T_s}(t) \exp(-jwt)\, dw && (1.101)\\
&= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t - nT_s) \exp(-jwt)\, dw && (1.102)\\
&= \sum_{n=-\infty}^{\infty} \exp(-jwnT_s) && (1.103)\\
&= \frac{2\pi}{T_s} \sum_{k=-\infty}^{\infty} \delta\left(w - k\frac{2\pi}{T_s}\right) && (1.104)
\end{aligned}$$

It is honest to admit that the last step in this derivation is far from trivial. The Fourier transform of an impulse train is also an impulse train but with an displacement in frequency between impulses that grows when the spacing in time decreases.

Therefore, the continuous Fourier transform of $\hat{f}$ can be written as:

$$\widehat{F}(w) = F(w) \circ \frac{2\pi}{T_s} \sum_{k=-\infty}^{\infty} \delta\left(w - k\frac{2\pi}{T_s}\right) = \frac{2\pi}{T_s} \sum_{k=-\infty}^{\infty} F\left(w - k\frac{2\pi}{T_s}\right) \quad (1.105)$$

where $F(w)$ is the Fourier transform of $f(t)$. This equation shows that $\widehat{F}(w)$ is build as an infinite sum of translated copies of $F(w)$. Each copies is centered on $k\frac{2\pi}{T_s}$. If T_s is small (i.e. if we sample very fast) then those copies will be far away from each other. But if we have few samples and T_s is large, those copies will get very close and will start mixing with each other. High frequency content in $F(w)$ will affect the low frequency content of $\widehat{F}(w)$, and this is exactly what produces aliasing. Figure 1.32 illustrates this. In this example, there is one band limited signal (i.e., there is a frequency, w_{max}, for which the magnitude of the Fourier transform is zero for all frequencies above w_{max}). First $T_s = 4$ seconds, in its FT we see replicates of the $F(w)$ centered around $\pi/2$. With $T_s = 8$ seconds, the replicates appear centered around $\pi/4$ and they start touching. $T_s = 8$ is slightly above the Nyquist's limit and some aliasing will exist. For $T_s = 16$ aliasing is severe and information will be lost making it impossible (without any additional prior information) to reconstruct the continuous function from its samples.

1.6.2 Reconstruction

If the copies do not touch, then we can see how it is possible to reconstruct the original continuous signal. We just need to apply a filter that has a constant gain for all the frequencies inside $w \in [-w_{max}, w_{max}]$, and 0 outside. The phase of the filter should be zero. This

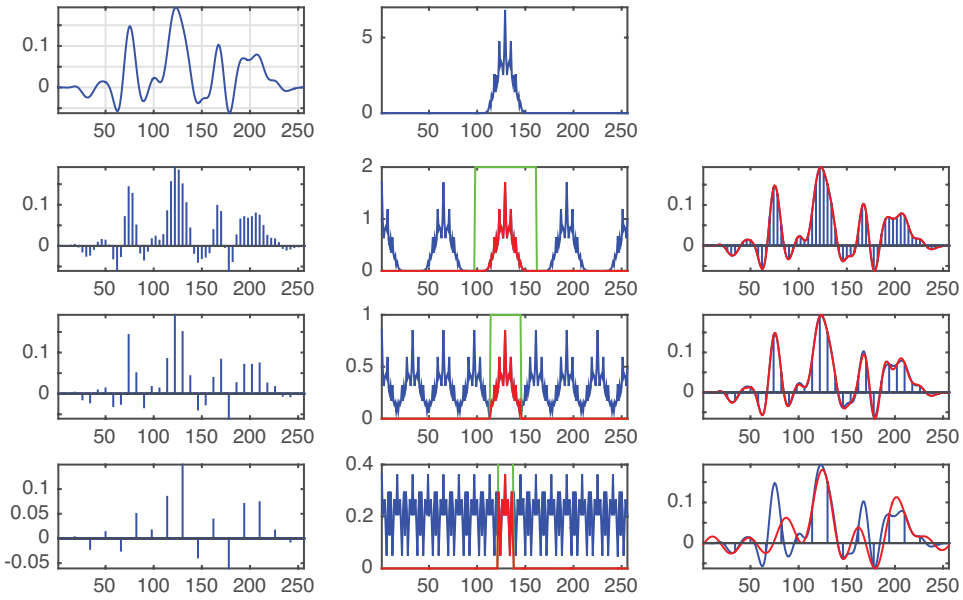


Figure 1.32

Aliasing examples. (a) - (f) Far left column: spatial sampling pattern. 2nd column: Fourier transform of that spatial pattern, revealing replication locations of the Fourier transform spectrum of the subsampled image. The subsampled image is shown in the 3rd column. Zeroing out all but the central replication of the image spectrum (far right), yields the interpolated images of the 4th column.

is:

$$H(w) = \begin{cases} \frac{T_s}{2\pi} & \text{if } w \in [-w_{max}, w_{max}] \\ 0 & \text{otherwise} \end{cases} \tag{1.106}$$

One piece of bad news: for any time limited signal (i.e., a signal that is defined inside an interval $t \in [a, b]$ and it is zero outside) the Fourier transform is not band limited. In other words, a signal can not be simultaneously time limited and band limited. Anyway, when something is impossible, generally it is because it does not matter and it might just mean that it is not the right way of thinking about the problem. So let's not worry about it.

The impulse response of such a filter is:

$$h(t) = \frac{\sin(t)}{t} = sinc(t) \tag{1.107}$$

it is called the sinc function.

In fact, it is easy to show that, in the lack of any other prior information, this is the optimal reconstruction in terms of the L2 norm. This is:

$$sinc(t) = argmin_h \int \left(f(t) - \hat{f}(t) \circ h(t)\right)^2 dt = argmin_H \int \left(F(w) - \widehat{F(w)}H(w)\right)^2 dw \tag{1.108}$$

then the function, $\widetilde{f}(t)$, that better reconstructs the input signal from its samples is:

$$\widetilde{f}(t) = \hat{f}(t) \circ sinc(t) = \sum_{n=-\infty}^{\infty} f\left[nT_s\right] sinc\left(\frac{t - nT_s}{T_s}\right) \tag{1.109}$$

where $\widetilde{f}(t)$ is the reconstructed signal and $\hat{f}(t)$ is the sampled signal. One disadvantage of this reconstruction is that the *sinc* function has infinite support which means that to interpolate each instant, we need to linearly combine all the samples $f\left[nT_s\right]$. Sometimes it is better to have a local reconstruction that only depends on the nearby samples. Indeed, there are other possible reconstructions that are not optimal in terms of L2 norm, but that only require local computations: linear, bilinear, bicubic, splines, etc. All of them can be written as a linear convolution with a kernel $h(t)$. In the case of the linear interpolation, the kernel $h(t)$ is a triangle of width $2T_s$.

1.6.3 2D spatial sampling

Let's now analyze what happens when sampling 2D signals to form discrete images.

In 2D things get more interesting. If we have a continuous image $f(x, y)$ we can sample it using a rectangular grid as $f\left[n, m\right] = f\left(nT_x, mT_y\right)$. We can do a very similar analysis to the one we just did for the 1D case. But in 2D we can have more interesting sampling patterns. For instance, we could define the discrete image as:

$$f\left[n, m\right] = f\left(an + bm, cn + dm\right) \tag{1.110}$$

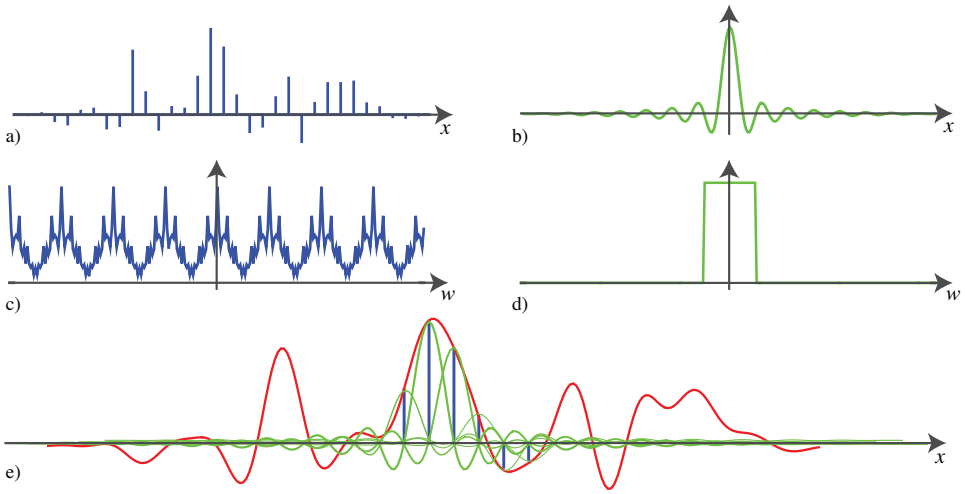


Figure 1.33

Reconstruction. a) Signal multiplied by a delta train. Each line corresponds to one impulse. The height of each impulse corresponds to the value of its integral. b) sinc function. c) magnitude of the Fourier transform of (a). d) Fourier transform of (b). The width of the box is set to cover just the central repetition of the FT shown in (c). e) Illustration of the reconstruction process. The sinc functions are scaled and shifted on top of each sample and then summed up (only six are shown). Note how the zero crossings coincide with the sample locations. The sum of all the sinc function corresponds to the red curve.

where a, b, c, d are constants. For instance, if $a = T, b = 0, c = 0, d = T$ then we will have a regular rectangular sampling. But we could have other patterns. For instance, if we set $a = T_1, b = -T_2/2, c = 0, d = T_2$ then we obtain an hexagonal sampling. So, now we can ask the following question: what is the optimal 2D sample arrangement given a fixed number of samples? The answer will require studying how aliasing will happen. What we want is to chose the sample arrangement that will allow the best reconstruction of the input continuous signal from a fixed number of samples. As we did with the 1D case, we can address this by studying the relationship between the Fourier transform of the continuous signal and the sampled one.

$$\widehat{f}(x, y) = f(x, y) \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \delta(x - an - bm, y - cn - dm) \tag{1.111}$$

Where the 2D delta train can be written using vector notation for the continuous spatial coordinates:

$$\delta_A(\vec{x}) = \sum_{\vec{n} \in \mathbb{Z}^2} \delta(\vec{x} - A\vec{n}) \tag{1.112}$$

where $\vec{x} = (x, y)^T$, $\vec{n} = (n, m)^T$, and A is the matrix:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{1.113}$$

The continuous Fourier transform of this delta train can be done by applying a change in variables and then using a similar procedure as the one followed in the 1D case. The result is:

$$\Delta_A(\vec{w}) = \frac{(2\pi)^2}{|A|} \sum_{\vec{k} \in \mathbb{Z}^2} \delta\left(\vec{w} - 2\pi A^{-1}\vec{k}\right) \tag{1.114}$$

Therefore, the Fourier transform of the sampled signal $\widehat{f}(x, y)$ is:

$$\widehat{F}(\vec{w}) = \frac{(2\pi)^2}{|A|} \sum_{\vec{k} \in \mathbb{Z}^2} F\left(\vec{w} - 2\pi A^{-1}\vec{k}\right) \tag{1.115}$$

Remember that for 2×2 matrices the inverse is easy to write:

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{1.116}$$

We can now check what happens with different sampling strategies. For the 2D rectangular sampling, eq. 1.115, simplifies to:

$$\widehat{F}\left(w_x, w_y\right) = \frac{(2\pi)^2}{T^2} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} F\left(w_x - \frac{2\pi}{T}k_1, w_y - \frac{2\pi}{T}k_2\right) \tag{1.117}$$

This is similar to the 1D case. Figure 1.34 shows two different delta trains for two different sampling patterns and also their Fourier transforms. The region delimited by the green polygon shows the region of valid frequencies. If the input signal only has spectral content within that region, then there will be no aliasing. The optimal sampling will be the one that makes that region as large as possible for a fixed number of samples. The optimal sampling strategy is the regular hexagonal sampling. This is not the sampling used in computer vision as all images are always represented on a rectangular grid, but an hexagonal sampling achieves an increase of around 10% in resolution for the same amount of samples. In fact, the distribution of photoreceptors in the eye [**?**] are distributed on an hexagonal array as shown in figure 1.35. Working with convolutional filters defined over an hexagonal grid is more efficient and it can achieve better radial symmetry [**?**].

For all the examples and derivations in this book, we will be working always of a regular rectangular grid.

1.6.3.1 Aliasing and anti-aliasing filter Sampling with the wrong frequency has interesting effects in 2D. Figure 1.36.a shows an example of a picture downsampled at different resolutions (412×512, 103×128, 52×64, and 26×32) and then reconstructed to the original resolution (412×512 pixels). For the figures, as we do not have access to the continuous image, we always work with sampled versions. But the original image is very high resolution and we can think of it as being the continuous image.

The images in Figure 1.36.a show the effects of aliasing. The stripes in the Zebra's body change orientation as we down sample them. And for the lowest resolution image, it is even hard to recognize the animal as being a zebra. Figure 1.36.b shows what happens with the image Fourier transform when we multiply it with the delta train (compare it with fig. 1.34). Figure 1.36.c shows the magnitude of the DFT of the sampled image (it corresponds to the region inside the green square in fig. 1.36.b). The DFT changes substantially, due to aliasing, from one resolution to the next one.

In order to reduce aliasing artifacts we need to filter the continuous signal with a low-pass filter in order to make it band-limited. Then we will be able to sample it avoiding high-spatial frequencies to interfere with the low-frequency content of the image. The anti-aliasing filter will not prevent from loosing the information contained in the high spatial frequencies. Figure 1.36.e shows the reconstructed images at different resolutions when an antialising filter is applied before sampling. Each resolution requires a different filter. The antialising filter can be a box filter like in eq. 1.106 with the support equal to the green region in Figure 1.36.b.

1.6.4 Spatio-temporal sampling

Spatio-temporal sampling can be studied with the tools we have already seen. In particular, eq. 1.115 can explain sampling in the N-dimensional case. Temporal aliasing is responsible

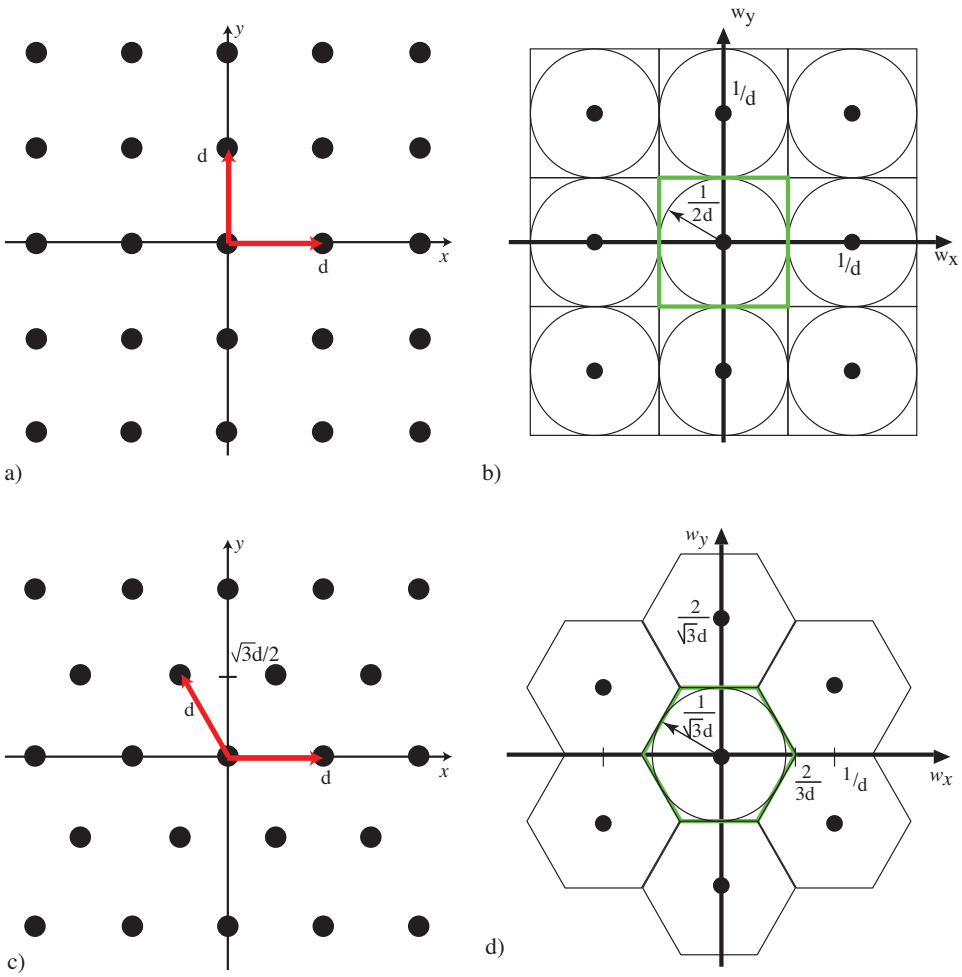


Figure 1.34
Sampling patterns.

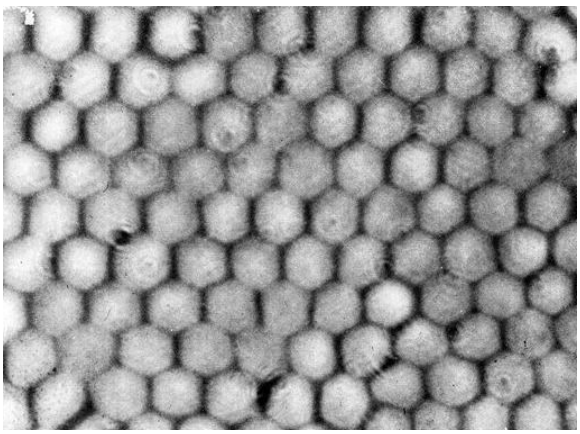


Figure 1.35
Distributions of cones in the fovea (cite source).

of the typical illusion in which we see wheels or fans changing the sense of rotation in movies. To avoid those artifacts it is important to apply an antialiasing filter as before.

There are many strategies commonly used to sample movies and each one has different advantages/disadvantages in terms of hardware implementation, efficiency, etc.

The most common type of spatiotemporal sampling is regular sampling: here, all the pixels are sampled in a regular spatial 2D grid and at regular time instants in which all the pixels are exposed simultaneously. This is also called Global Shutter mode.

In many cameras (DSLRs, mobile phones, ...) the most commonly used sampling is the Rolling Shutter mode. Here, every row in the image is sampled simultaneously. But different rows are sampled at different instants. This sampling mode allows for a faster sampling rate with current hardware implementations but it can create spatial distortions in the image if the camera moves or when taking pictures of moving objects.

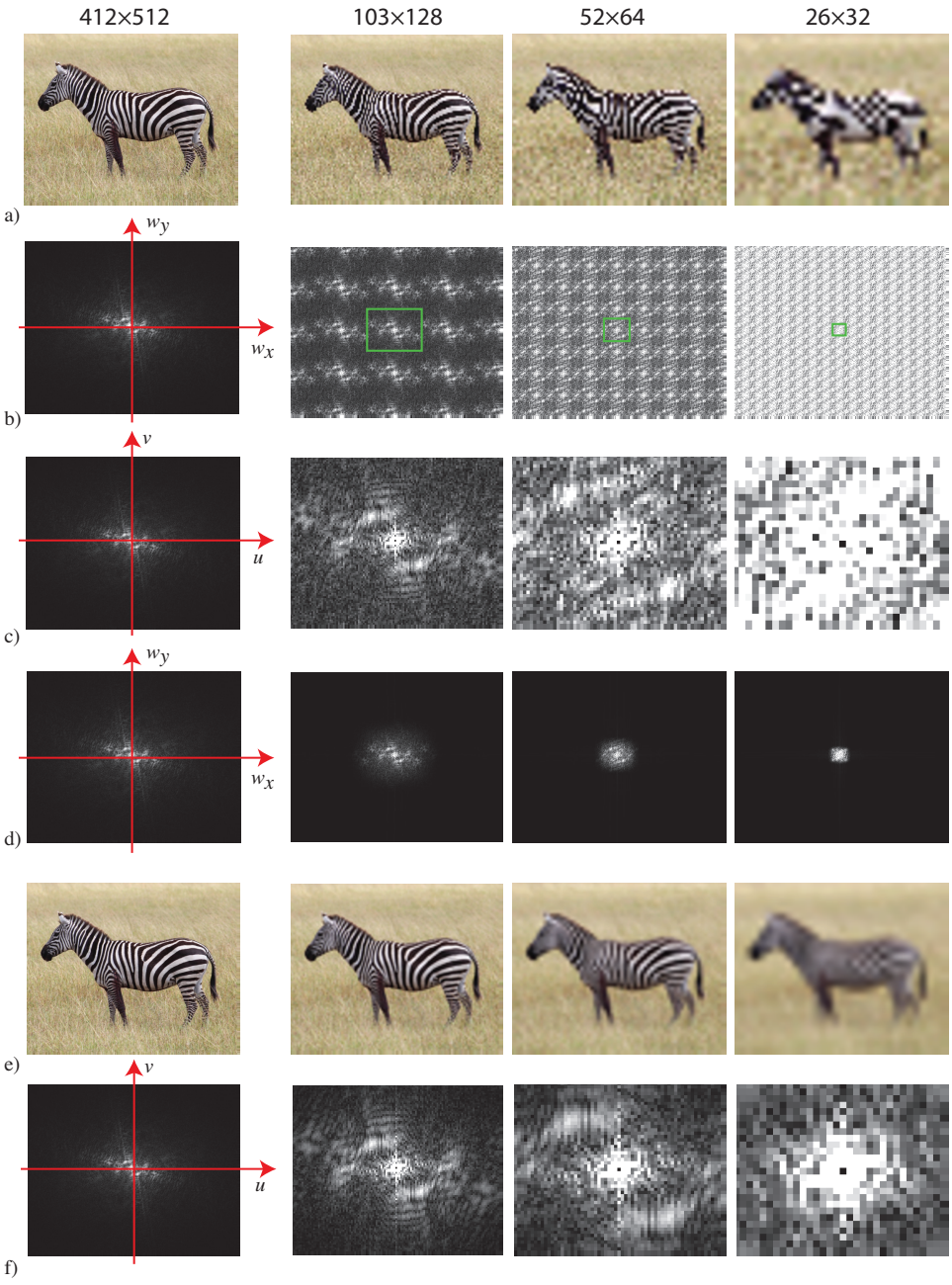


Figure 1.36

Aliasing and antialiasing filter. a) The zebra sampled with aliasing starts looking as a cow. b) Fourier transform of the continuous signal $f(x, y)$ multiplied by delta trains: $\widehat{f(x, y)}$, c) Discrete Fourier transform of the corresponding sampled signals, $f[n, m]$, and d) Fourier transform of the reconstructed signal. e) Sampled image after processing it with an antialiasing filter. f) Discrete Fourier transform of the corresponding antialiased sampled images, $f[n, m]$. Note that now the central part of the Fourier transform is not changing.

Bibliography

[1] Adelson, Edward H. 2000. Lightness perception and lightness illusion. In *The new cognitive neurosciences*, ed. M. Gazzaniga, 339–351.

[2] Adelson, Edward H., and James R. Bergen. 1991. The plenoptic function and the elements of early vision. In *Computational models of visual processing*, eds. Michael S. Landy and Anthony J. Movshon, 3–20. Cambridge, MA: MIT Press.

[3] Cavanagh, P. 1996. Vision is getting easier every day. *Perception* 24: 1227–1232.

[4] Papert, Seymour. 1966. The summer vision project, MIT AI Memo 100, Massachusetts Institute of Technology, Project Mac.

[5] Roberts, Lawrence G. 1963. *Machine perception of three-dimensional solids*. *Outstanding dissertations in the computer sciences*. Garland Publishing, New York.