Contents

Chapter 1

Linear image filtering

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: a linear filter. 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). In this chapter we will see how far it takes us toward these goals.

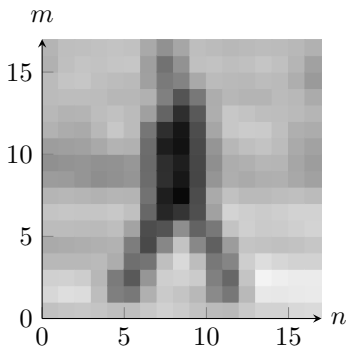
1.1 Signals and systems

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.

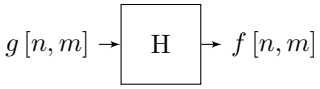
Gray scale images are two dimensional signals that can be encoded as arrays of pixels: $\mathbf{I}[n, m]$, where n and m index the vertical and horizontal dimensions. Can you guess what object appears in the following image?

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

This matrix represents the same image of 18×18 pixels as encoded in a computer. The goal of a vision system is to reorganize the array of numbers into a coherent representation of the scene depicted in the image:



The kind of systems that we will study here take an image, $g[n,m]$, as input, perform some filtering operation, H, and output another image, $f[n,m] = H(g[n,m])$



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. The generality of non-linear filters makes very difficult to characterize them. Therefore, we will start with simpler family of filters.

1.2 Linear Filters

Among all possible filters, the simplest ones are linear filters. They represent a very small portion of all the possible filters one could implement, but we will see that they 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:

$$f[n] = \sum_{k=0}^{N-1} h[n,k]\, g[k] \quad for\ n \in [0, M-1] \tag{1.1}$$

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:

$$\begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M-1] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[0,1] & ... & h[0,N-1] \\ h[1,0] & h[1,1] & ... & h[1,N-1] \\ \vdots & \vdots & \vdots & \vdots \\ h[M-1,0] & h[M-1,1] & ... & h[M-1,N-1] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[N-1] \end{bmatrix} \tag{1.2}$$

which we will write as

$$f = Hg \tag{1.3}$$

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

Figure 1.1: A fundamental property of images is translation invariance–the same image may appear at arbitrary spatial positions within the image. Image credit: Fredo Durand.

output image is $f[n,m]$, and the input image is $g[n,m]$, then a general linear filtering of the image is

$$f[n,m] = \sum_{k,\ l=0}^{N-1,\ M-1} h[n,m,k,l]\, g[k,l] \tag{1.4}$$

By writing the images as column vectors, concatenating all the image columns into a long vector, we can also write the previous equation using matrices and vectors: $f = Hg$.

1.3 Convolution and translation invariant filtering

General linear filters are still too general for us. So let's consider an even smaller family of filters: translation invariant linear filters. Translation invariant filters can be motivated by the following observation: typically, we don't know where within the image we expect to find any given item (Fig. 1.1), so we often want to process the image in a spatially invariant manner, the same processing algorithm at every pixel. In that case, the processing becomes a *linear convolution* of the image data with some filter.

Linear translation invariance imposes a strong constraint on the form of equation 1.1. 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:

$$f[n] = h \circ g = \sum_{k=0}^{N-1} h[n-k]\, g[k] \tag{1.5}$$

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 $n \in [0, N-1]$.

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.

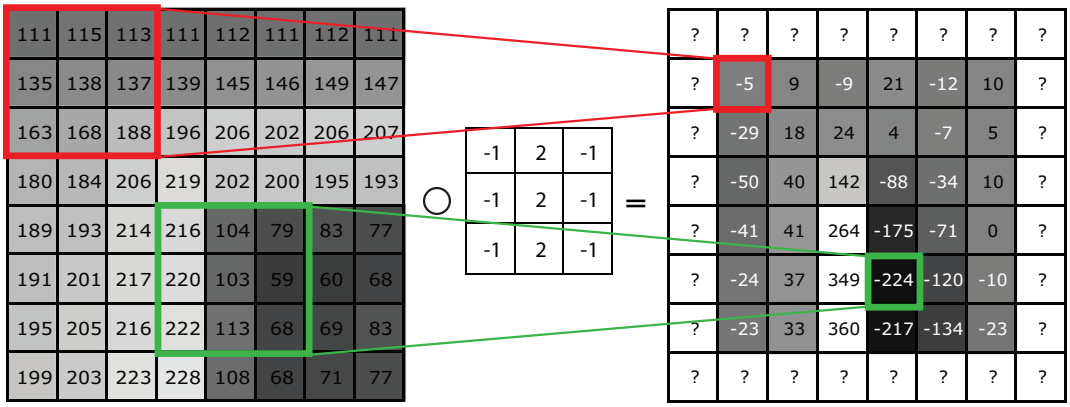


Figure 1.2: Illustration of a 2-d convolution of an input image, g, convolved with a kernel, h, giving the output image, f. The images are shown with both their pixel values and the corresponding image intensities (the assignment of intensities to numbers was rescaled for the output image, f). Border pixel values of the output image are not determined by the convolution, since the kernel would include pixel values outside of the input image.

Mathematically, this is:

$$f[m,n] = h \circ g = \sum_{k,l} h[m-k, n-l]\, g[k,l] \tag{1.6}$$

Figure 1.2 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 1.3 shows several simple convolution examples. Figure 1.3.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 1.3.b shows a kernel that produces a shift of the input image. For the last example shown in figure 1.3.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 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. 1.5 or equation 1.6 is it easy to show that convolution is commutative operator:

$$h[n] \circ g[n] = g[n] \circ h[n] \tag{1.7}$$

  this means that the order of convolutions is irrelevant.

- It is associative:

$$h[n] \circ g[n] \circ q[n] = h[n] \circ (g[n] \circ q[n]) = (h[n] \circ g[n]) \circ q[n] \tag{1.8}$$

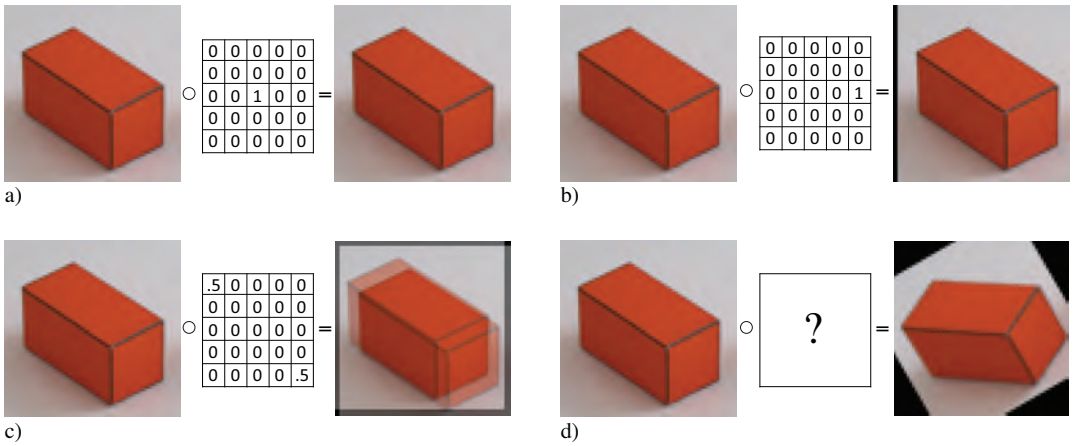


Figure 1.3: a) An impulse convolved with the input image gives no change (each color channel is convolved with the same kernel). b) A shifted impulse shifts the image. c) Sum of two shifted copies of the image. d) The text discusses why there is no space invariant convolution kernel can rotate an image. All the examples use zero padding for handling boundary conditions.

- It is distributive with respect to the sum:

$$h[n] \circ (f[n] + g[n]) = h[n] \circ f[n] + h[n] \circ g[n] \tag{1.9}$$

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

$$f[n - n_0] = h[n] \circ g[n - n_0] = h[n - n_0] \circ g[n] \tag{1.10}$$

- 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, the impulse: $\delta[n]\,[1, 0, 0, ..., 0]$, it takes the value 1 for $n = 0$ and it is zero everywhere else.

1.3.2 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. 1.4):

- 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.

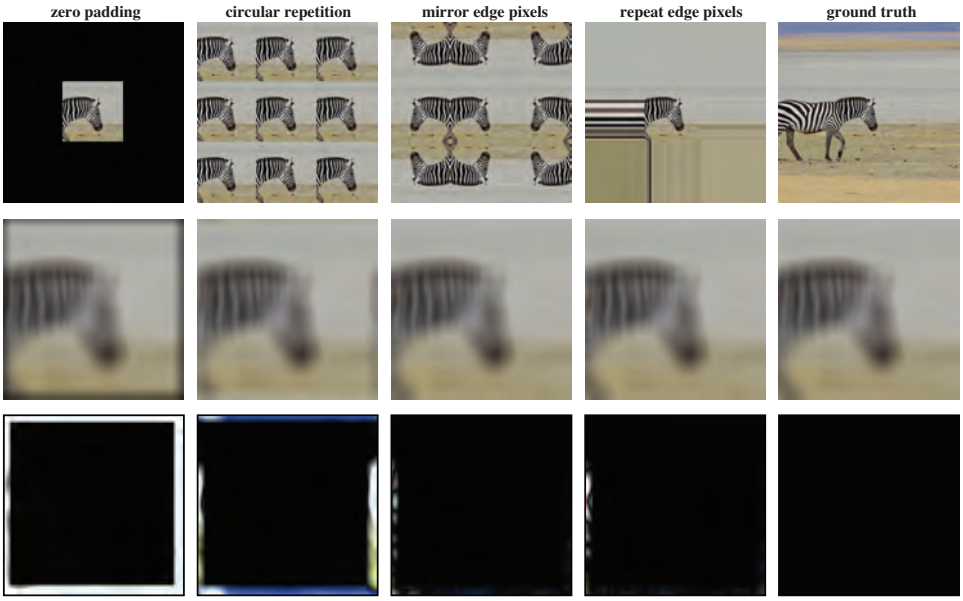


Figure 1.4: Boundary extensions are different ways of approximating the ground truth image that exists beyond the image boundary. Each column shows: a) Different types of boundary extension. The last image shows the ground truth. b) the output of convolving the image with a kernel that is a box of 1 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.

- 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.

1.4 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.

1.4.1 Correlation vs. convolution

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

$$f[m, n] = h \circ g = \sum_{k,l} h[m - k, n - l]\, g[k, l] \tag{1.11}$$

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:

$$f[m, n] = h * g = \sum_{k,l} h[m + k, n + l]\, g[k, l] \tag{1.12}$$

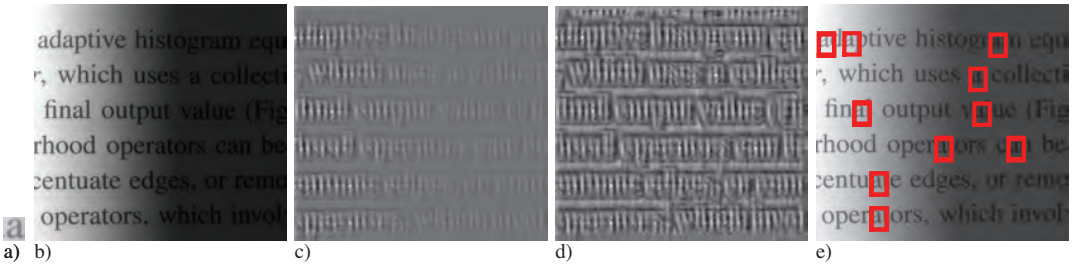


Figure 1.5: a) Template. b) Input image. c) Output of the correlation between the image and the template. d) Output of the normalized correlation. e) Locations with values of the normalized correlation above 75% of its maximum value.

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.

1.4.2 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 1.5 illustrates the detection of the "a" letters through matching an example "a" template (figure 1.5(a)) in a given text image (figure 1.5(b)).

Although correlation is a potential method for template matching it also has certain flaws. Consider matching the "a" template in figure 1.5(a) in a given input image. Just by increasing the brightness of the image in different parts we can increase the filter response 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:

$$f' = f - m_f \quad \text{where} \quad m_f = \frac{1}{MN} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \tag{1.13}$$

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:

$$NCC(f,g) = \langle \bar{f}, \bar{g} \rangle \quad \text{where} \quad \bar{f} = \frac{f - m_f}{\sqrt{\langle f - m_f, f - m_f \rangle}} = \frac{f'}{\sqrt{\langle f, f' \rangle}},$$
$$\bar{g} = \frac{g - m_g}{\sqrt{\langle g - m_g, g - m_g \rangle}} = \frac{g'}{\sqrt{\langle f', f' \rangle}} \tag{1.14}$$

Note that both $\bar{f}$ and $\bar{g}$ are unit norm vectors. Therefore $NCC(f,g) = \langle \bar{f}, \bar{g} \rangle = ||\bar{f}||\,||\bar{g}||\,cos\alpha = cos\alpha$ 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:

$$SSD(f,g) = ||f - g||^2 = \sum_n |f[n] - g[n]|^2 = E_f + E_g - 2\langle f, g \rangle \tag{1.15}$$

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:

$$SSD(\hat{f}, \hat{g}) = ||\hat{f} - \hat{g}||^2 = E_{\hat{f}} + E_{\hat{g}} - 2\langle f, g \rangle = 2 - 2\left\langle \hat{f}, \hat{g} \right\rangle \tag{1.16}$$

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:

$$SSD(\hat{f}, \hat{g}) = 2 - 2cos\theta \quad \text{since} \quad \left\langle \hat{f}, \hat{g} \right\rangle = ||\hat{f}||\,||\hat{g}||\,cos\theta = cos\theta \tag{1.17}$$

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:

$$cos_{sim}(f, g) = \frac{\langle f, g \rangle}{||f||\,||g||} \tag{1.18}$$

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 (1.13).

Convolution and correlation operators are the main building blocks of the convolutional neural networks.

Chapter 2

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.

2.1 Image transforms

Sometimes it is useful to transform the image pixels into another representation that might reveal image properties that can be useful for solving vision. tasks. Before we saw that linear filtering is a useful way of transforming an image. Linear image transforms with the form $g = Hf$ where H is a matrix of size $N \times N$, can be thought as a way of changing the initial pixels representation of f into a different representation in g. This representation is specially interesting when it can be inverted so that the original pixels can be recovered as $f = H^{-1}g$. The Fourier transform is. one of those representations. It has the advantage that the transformed signal g has a number of interesting properties not immediately available in the original pixels. Fourier transforms are the basis of a number of computer vision approaches and are an important tool to understand images and how linear spatially invariant filters transform images.

2.2 Sines, cosines and complex exponentials

Let's start by defining two very useful image families: the discrete sine and cosine waves. They are defined as:

$$s_{u,v}[n,m] = A\sin\left(2\pi\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{2.1}$$

$$c_{u,v}[n,m] = A\cos\left(2\pi\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{2.2}$$

where u and v are the two spatial frequencies and define how fast or slow the waves change along the spatial dimensions n and m. Figure 2.1 shows some examples.

Another important signal is the complex exponential wave. In 2D, the complex exponential wave is:

$$e_{u,v}[n,m] = \exp\left(2\pi j\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{2.3}$$

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:

$$e_{u,v}[n,m] = e_u[n]\,e_v[m] \tag{2.4}$$

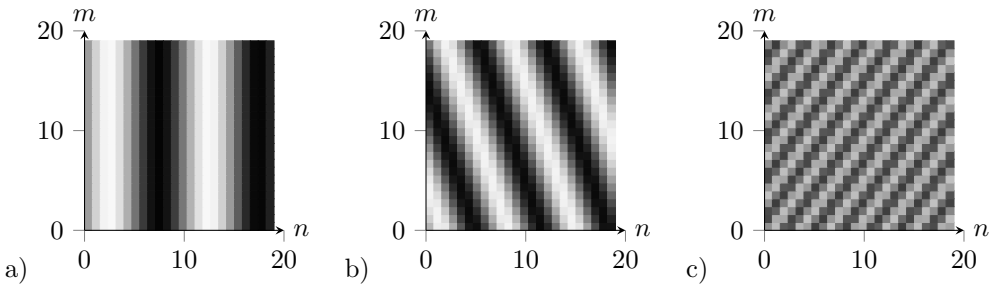


Figure 2.1: 2D sine waves with $N = M = 20$. The frequency values are: a) $u = 2, v = 0$, b) $u = 3, v = 1$, c) $u = 7, v = -5$

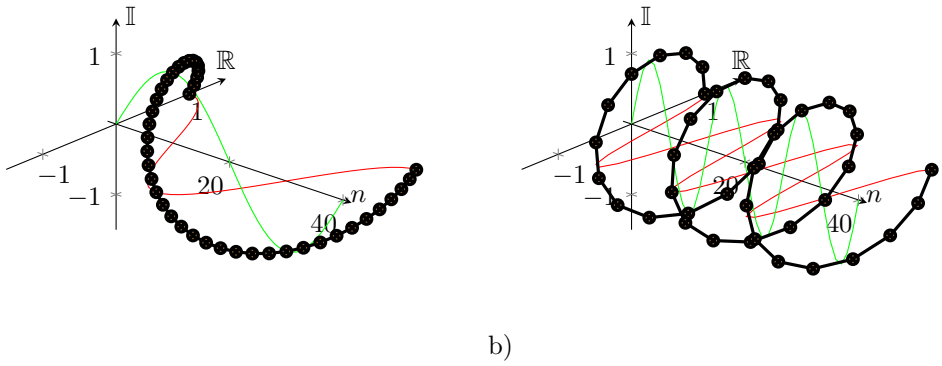


Figure 2.2: Complex exponential wave with a) $N = 40$, $k = 1$, $A = 1$, and b) $N = 40$, $k = 3$, $A = 1$. The red and green curves show the real and imaginary waves. The yellow line is the complex exponential. The dots correspond to the discrete samples.

Figure 2.2 shows the discrete complex exponential function (for $v = 0$). 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 remarkable property, the 2D complex exponentials form an orthogonal basis for discrete images of size $N \times M$. In fact,

$$\langle e_{u,v}, e_{u',v'} \rangle = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} e_{u,v}[n,m]\, e^*_{u',v'}[n,m] = MN\delta[u-u']\,\delta[v-v'] \tag{2.5}$$

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

2.3 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:

$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{2.6}$$

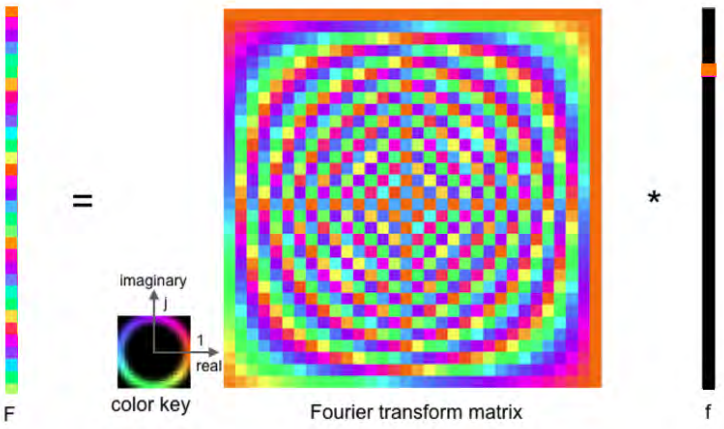


Figure 2.3: Visualization of Discrete Fourier Transform as a matrix. The signal to be transformed forms the entries of the column vector at right. The complex values of the Fourier Transform matrix are indicated by the color, with the key in the bottom left. In the vector at the right, black values indicate zero.

By applying $\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}$ to both sides of Eq. (2.6) and exploiting the orthogonality between distinct Fourier basis elements, we find the inverse Fourier transform relation:

$$f[n,m] = \frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} F[u,v]\exp\left(+2\pi j\left(\frac{u\,n}{N}+\frac{v\,m}{M}\right)\right) \tag{2.7}$$

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 2.8 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 2.3 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 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:

$$f[n,m] = \frac{1}{NM}\sum_{u=-N/2}^{N/2}\sum_{v=-M/2}^{M/2} F[u,v]\exp\left(+2\pi j\left(\frac{u\,n}{N}+\frac{v\,m}{M}\right)\right) \tag{2.8}$$

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

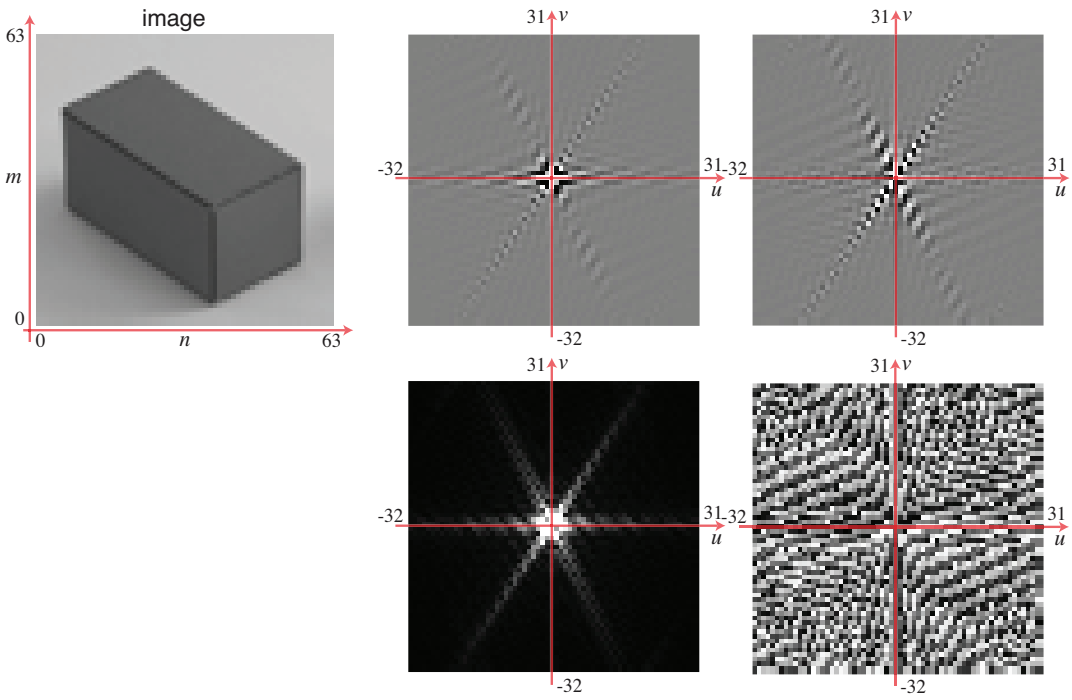


Figure 2.4: DFT of an image.

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.

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.

2.4 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 2.4 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:

$$F[u,v] = Re\{F[u,v]\} + j\, Imag\{F[u,v]\} \tag{2.9}$$

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

$$F[u,v] = A[u,v]\, \exp(j\,\theta[u,v]) \tag{2.10}$$

where $A[u,v] \in \mathbb{R}^+$ is the amplitude and $\theta[u,v] \in [-\pi, \pi]$ is the phase. Figure 2.4 shows both decompositions of the Fourier transform.

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 2.5 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.

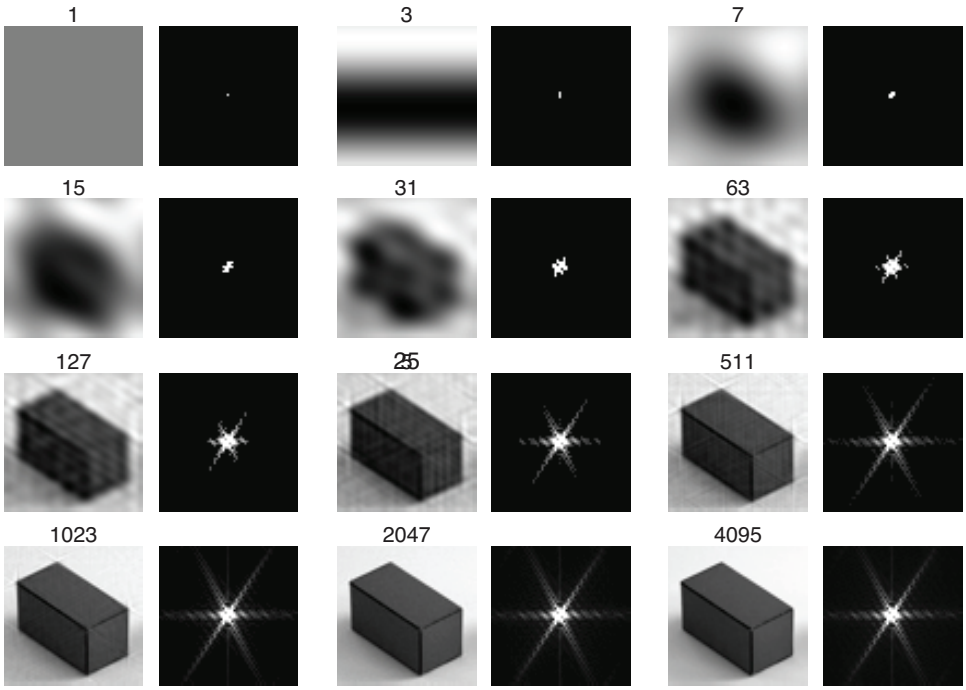


Figure 2.5: Reconstructing an image from the N Fourier coefficients of the largest amplitude. The right frame shows the location, in the Fourier domain, of the N Fourier coefficients which, when inverted, give the image at the left.

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.

2.5 Useful transforms

It's useful to become adept at computing and manipulating simple Fourier transforms. For some simple cases, we can compute the analytic form of the Fourier transform.

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

$$F[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} \delta[n,m] \exp\left(-2\pi j\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) = 1 \tag{2.11}$$

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:

$$\delta[n,m] = \frac{1}{NM}\sum_{u=-N/2}^{N/2}\sum_{v=-M/2}^{M/2} \exp\left(2\pi j\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) \tag{2.12}$$

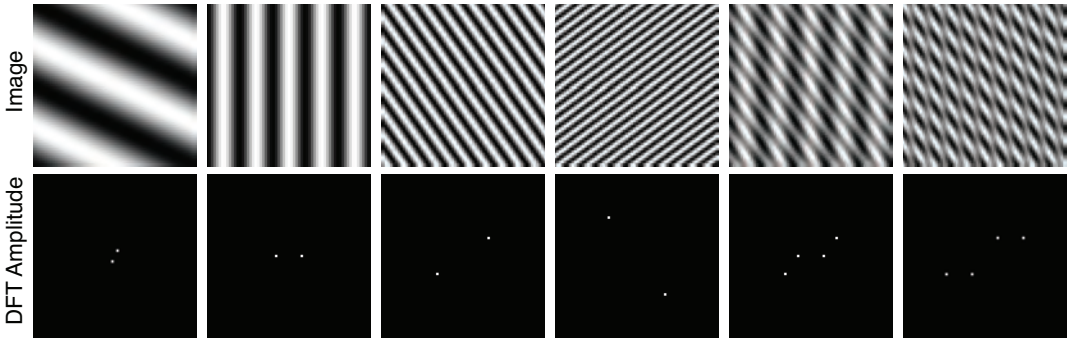


Figure 2.6: Some two-dimensional Fourier transform pairs. Images are 64×64 pixels. The waves are *cos* with frequencies $(1, 2)$, $(5, 0)$, $(10, 7)$, $(11, -15)$. The last two examples show the sum of two waves and the product.

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:

$$F[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} \cos\left(2\pi\left(\frac{u_0\,n}{N} + \frac{v_0\,m}{M}\right)\right) \exp\left(-2\pi j\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) = \quad (2.13)$$

$$= \frac{1}{2}\left(\delta[u - u_0, v - v_0] + \delta[u + u_0, v + v_0]\right) \quad (2.14)$$

this can be easily proven using Euler's equation **??** 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:

$$F[u,v] = \frac{1}{2j}\left(\delta[u - u_0, v - v_0] - \delta[u + u_0, v + v_0]\right) \quad (2.15)$$

Figure 2.6 shows the DFT of several waves with different frequencies and orientations.

Figure 2.7 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. 2.7, 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 **??**.

2.6 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:

$$DFT\{\alpha f[n,m] + \beta g[n,m]\} = \alpha F[u,v] + \beta G[u,v] \quad (2.16)$$

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: $F[u,v] = F_1[u]\,F_2[v]$

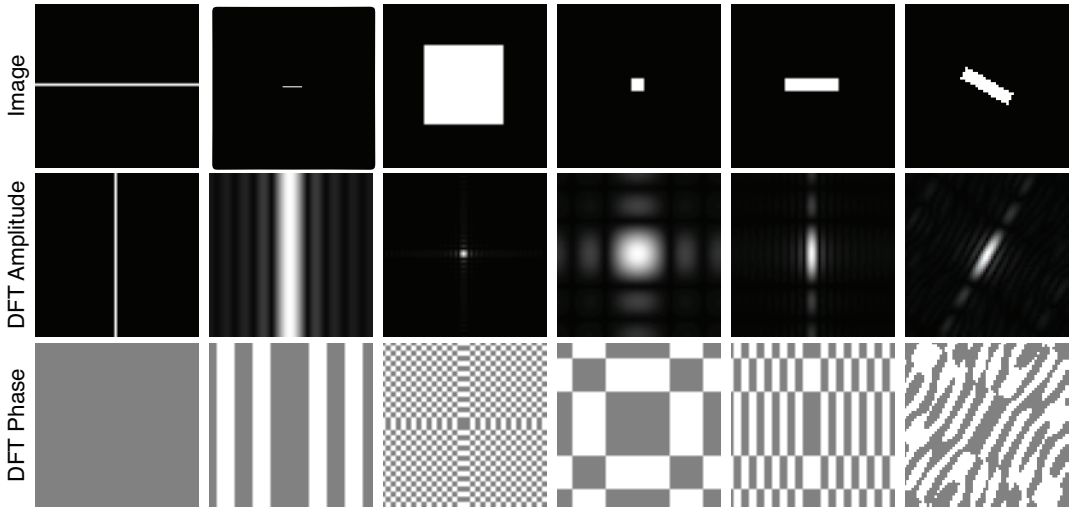


Figure 2.7: Some two-dimensional Fourier transform pairs. Note the trends visible in the collection of transform pairs: As the support of the image in one domain gets larger, the magnitude in the other domain becomes more localized. A line transforms to a line oriented perpendicularly to the first. Images are 64×64 pixels.

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

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:

$$\begin{aligned} DFT\{f[n-n_0, m-m_0]\} = \\ &= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n-n_0, m-m_0] \exp\left(-2\pi j\left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) = \\ &= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n, m] \exp\left(-2\pi j\left(\frac{u\,(n+n_0)}{N} + \frac{v\,(m+m_0)}{M}\right)\right) = \\ &= F[u, v] \exp\left(-2\pi j\left(\frac{u\,n_0}{N} + \frac{v\,m_0}{M}\right)\right) \end{aligned} \tag{2.17}$$

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. 2.8 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. 2.17 will not apply exactly. To verify eq. 2.17 let's look at the real part of the DFT of each image shown in fig. 2.8.b and d. If eq. 2.17 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. 2.8.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. 2.8.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. 2.8.f can be used to estimate the displacement between

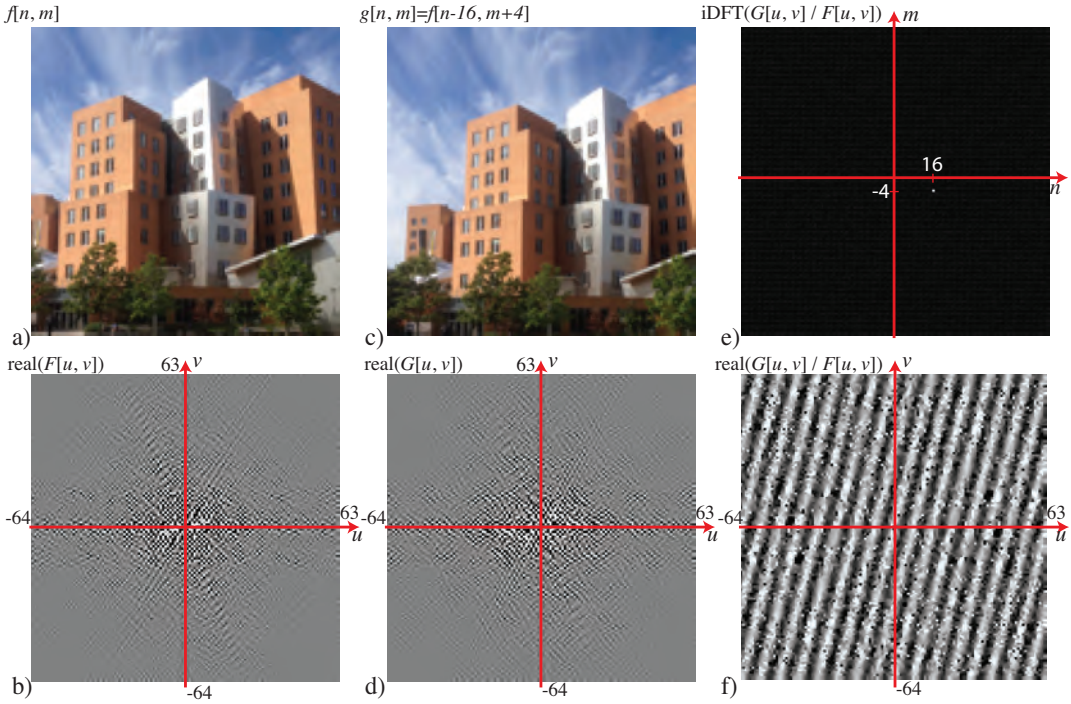


Figure 2.8: Translation in space. Image (c) corresponds to image (a) after a translation of 16 pixels to the right and 4 pixels down. Images (b) and (d) show the real parts of their corresponding DFTs (with $N = 128$). The image (f) shows the real part of the ratio between the two DFTs, and (e) is the 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.

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:

$$DFT\left\{f\left[n,m\right]\exp\left(-2\pi j\left(\frac{u_0\,n}{N}+\frac{v_0\,m}{M}\right)\right)\right\}=F\left[u-u_0,v-v_0\right] \tag{2.18}$$

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

A related relationship is:

$$DFT\left\{f\left[n,m\right]\cos\left(2\pi j\left(\frac{u_0\,n}{N}+\frac{v_0\,m}{M}\right)\right)\right\}=F\left[u-u_0,v-v_0\right]+F\left[u+u_0,v+v_0\right] \tag{2.19}$$

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:

$$\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}f\left[n,m\right]g^{*}\left[n,m\right]=\frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1}F\left[u,v\right]G^{*}\left[u,v\right] \tag{2.20}$$

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

$$\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left\|f\left[n,m\right]\right\|^{2}=\frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1}\left\|F\left[u,v\right]\right\|^{2} \tag{2.21}$$

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:

$$f=g\circ h \tag{2.22}$$

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

$$\begin{aligned}F\left[u,v\right]&=DFT\left\{g\circ h\right\}\\&=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}g\left[m-k,n-l\right]h\left[k,l\right]\exp\left(-2\pi j\left(\frac{mu}{M}+\frac{nv}{N}\right)\right)\end{aligned} \tag{2.23}$$

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

$$F[u,v] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} h[k,l] \sum_{m'=-k}^{M-k-1}\sum_{n'=-l}^{N-l-1} g[m',n'] \exp\left(-2\pi j\left(\frac{(m'+k)u}{M} + \frac{(n'+l)v}{N}\right)\right) \tag{2.24}$$

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

$$F[u,v] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} G[u,v] \exp\left(-2\pi j\left(\frac{ku}{M} + \frac{lv}{N}\right)\right) h[k,l] \tag{2.25}$$

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

$$F[u,v] = G[u,v]\, H[u,v] \tag{2.26}$$

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

$$f[n,m] = g[n,m]\, h[n,m] \tag{2.27}$$

is the convolution of their DFTs:

$$F[u,v] = \frac{1}{NM} G[u,v] \circ H[u,v] \tag{2.28}$$

2.7 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.

2.7.1 Amplitude and Phase

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

$$F[u,v] = A[u,v]\, \exp\left(j\,\theta[u,v]\right) \tag{2.29}$$

where $A[u,v] = |F[u,v]|$ and $\theta[u,v] = \angle F[u,v]$

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 2.9 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

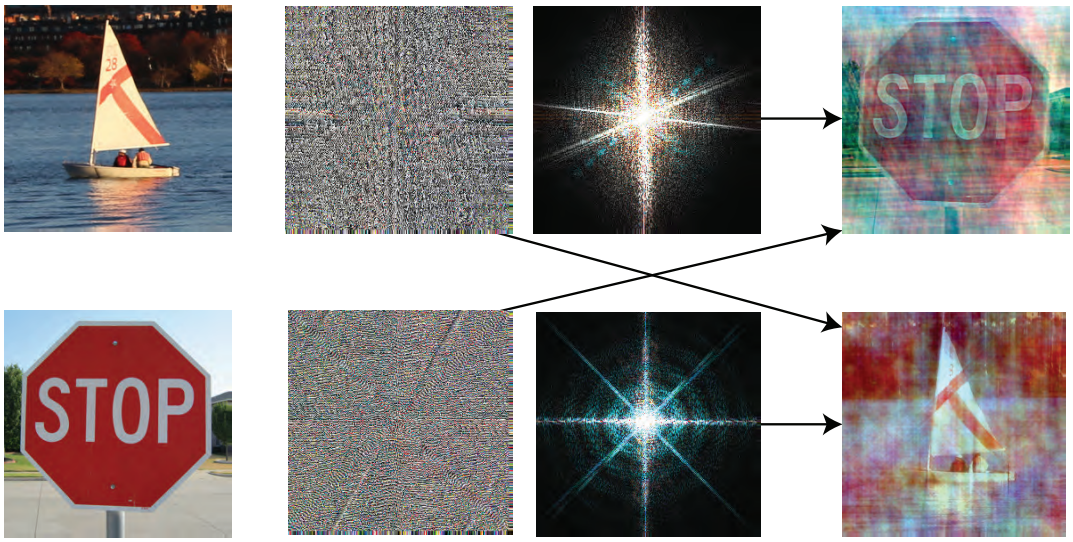


Figure 2.9: Swapping the amplitude and the phase of the Fourier Transform of two images. Each color channel is processed in the same way.

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 2.9 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.

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. 2.10. 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 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. 2.10 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. 2.9 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.

Take the following quiz: match the Fourier transform magnitudes with the corresponding images in Fig. 2.11

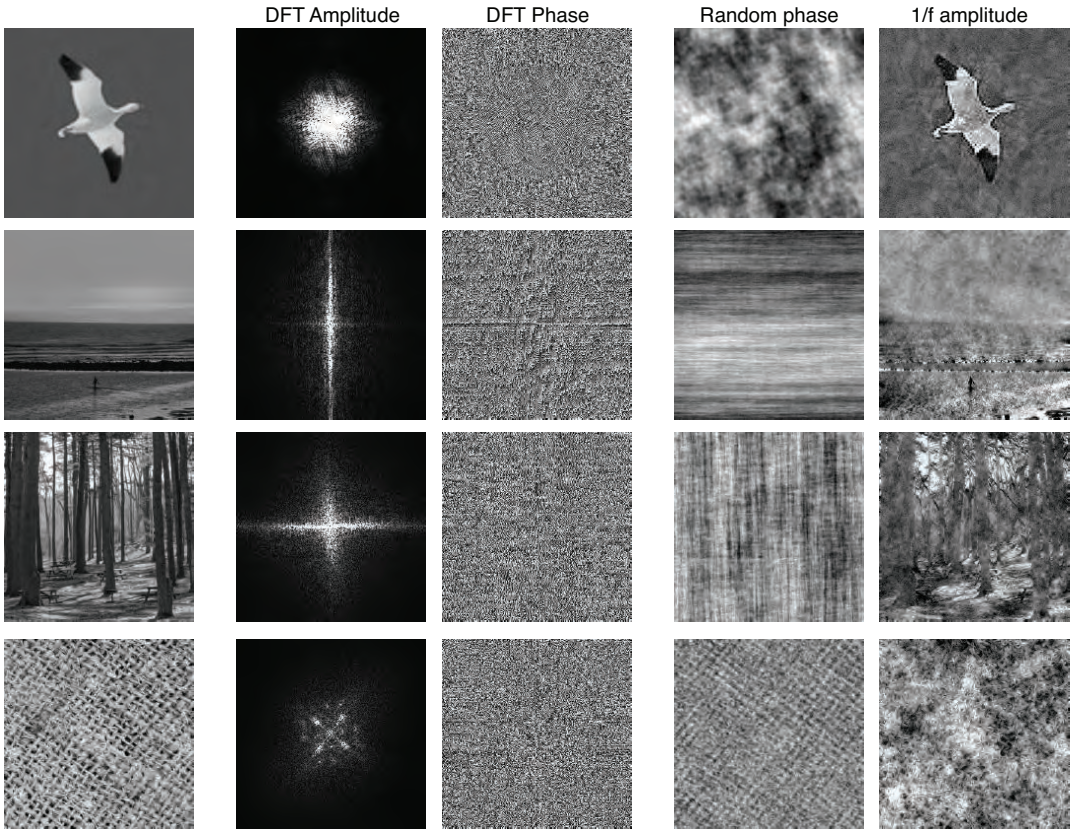


Figure 2.10: The relative importance of phase and amplitude depends on the image. Each row shows one image, its Fourier transform (amplitude and phase), and the resulting images obtained by applying the inverse Fourier transform to a signal with the original amplitude and randomized phase, and a signal 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.

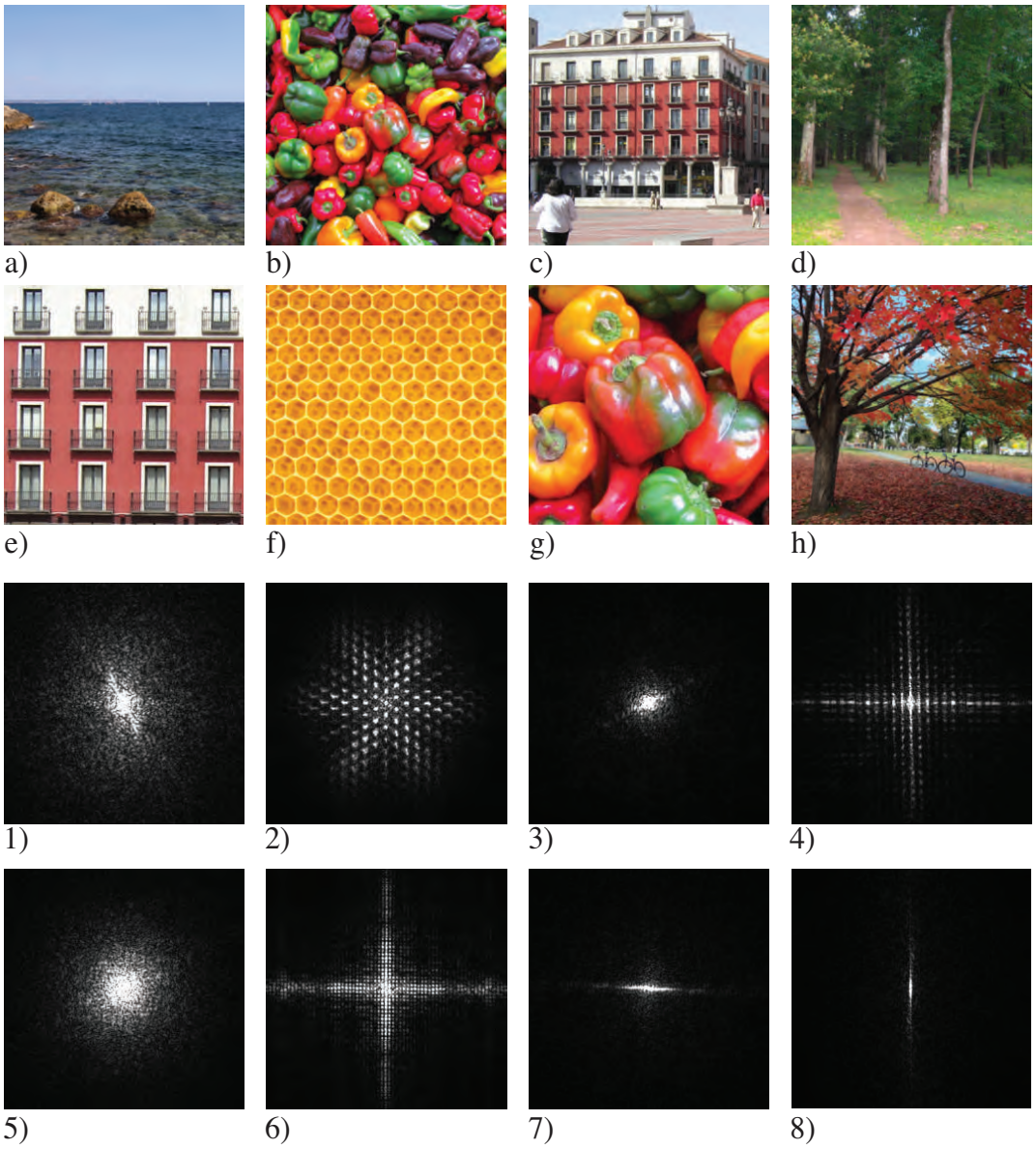


Figure 2.11: The Fourier transform matching game: Match each image (a-h) with its corresponding Fourier transform magnitude (1-8). The correct answer is: 1-h, 2-f, 3-g, 4-c, 5-b, 6-e, 7-d, 8-a.

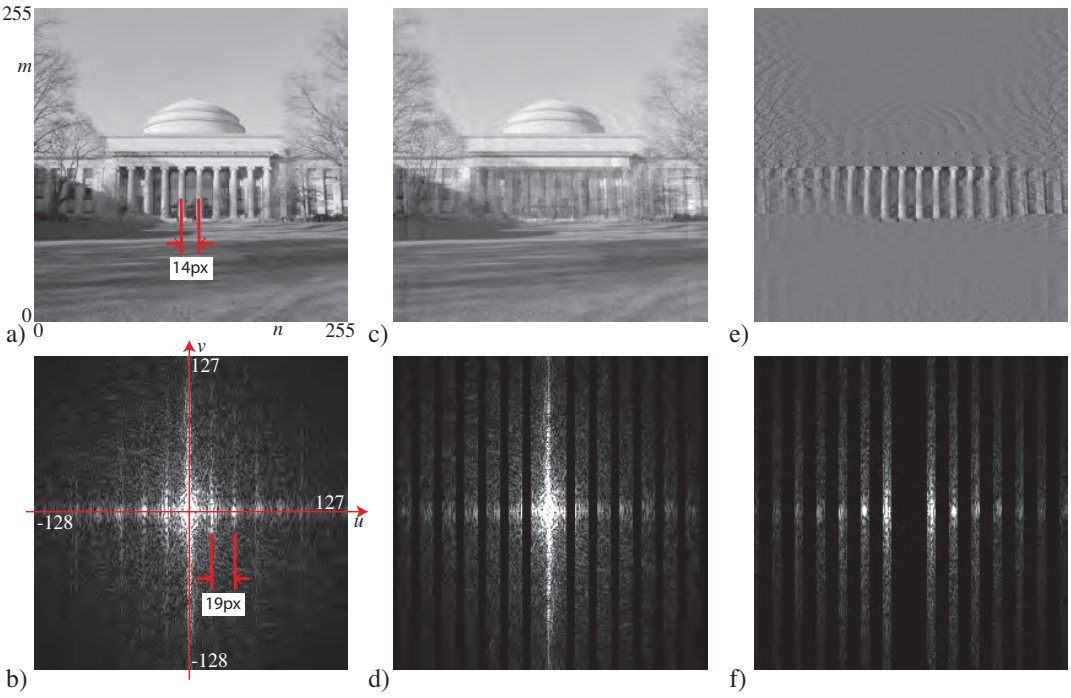


Figure 2.12: Simple filtering in the Fourier domain. (a) The repeated columns of the building of the MIT dome generate harmonics along a horizontal line in the Fourier domain. (b) By zeroing out those Fourier components, the columns of the building are substantially removed.

2.8 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. 2.12.a shows a picture of the main MIT building. The columns produce a quasi periodic pattern. Fig. 2.12.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. 2.12.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. 2.12.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.

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. 2.12.d. Fig. 2.12.c shows the resulting image where the columns are almost gone while the rest of the image is little affected. Fig. 2.12.e shows the complementary image (in fact $a = c + e$) and its DFT Fig. 2.12.f.

2.9 Fourier analysis of linear filters

Linear convolutions, despite their simplicity, are surprisingly useful for processing and interpreting images. It's often very useful to blur images, in preparation for subsampling or to remove noise, for example. Other useful processing includes edge enhancement and motion analysis.

From the previous section we know that we can write linear filters as convolutions:

$$f[n,m] = h[n,m] \circ g[n,m] \tag{2.30}$$

where $h[n,m]$ is the impulse response of the system. We can also write this as a product in the Fourier domain:

$$F[u,v] = H[u,v]\, G[u,v] \tag{2.31}$$

The function $H[u,v]$ is called the transfer function of the filter. If we use the polar form:

$$H[u,v] = |H[u,v]| \exp\left(j \angle H[u,v]\right) \tag{2.32}$$

The magnitude $|H[u,v]|$ is the amplitude gain, and the phase $\angle H[u,v]$ is the phase shift.

The Fourier domain shows that, in many cases, what a filter does is to block or let pass certain frequencies. Filters are many times classified according to the frequencies that they let pass through the filter (low, medium or high frequencies):

- Low-pass filter
- Band-pass filter
- High-pass filter

But this classification of a filter might not be appropriate in many cases. Some filters have their main effect over the phase of the signal and they are better understood in the spatial domain. In general, filters affect both the magnitude and the phase of the input signal.