Chapter 7

The Fourier transform

In chapter 3 we introduced the concept of image frequencies. We used images of sinusoids to show the relation between the frequency of the sinusoids and image detail: a high frequency corresponded to a high level of detail, and a low frequency to a low level of detail. This seems natural; the grey values along a line in an image will cycle between black and white. If we encounter an area with many details, then the cycling must be done fast (high frequency), while relatively smooth image areas require only slow cycling (low frequency).

In this chapter we will investigate further the relation between sinusoids of different frequencies and digital images.

7.1 The relation between digital images and sinusoids

In chapter 3, we compared image frequency to the frequency of sinusoids. But how can we determine the “frequency” of the grey values along a line in an image if the graph of these grey values looks nothing like a sinusoid? The answer can be gleaned from this statement: any digital image can be modeled as a sum of sinusoids. More general, any function \( f \) for which \( f \) and \( f'' \) are piecewise continuous can be modeled as a sum of sinusoids.

Example

The function \( f(x) = x \) on the domain \([−\pi/2, \pi/2]\) can be written as

\[
\begin{align*}
f(x) &= \sin(2x) - \frac{\sin(4x)}{2} + \frac{\sin(6x)}{3} - \frac{\sin(8x)}{4} + \frac{\sin(10x)}{5} - \frac{\sin(12x)}{6} + \ldots \\
&= \sum_{j=1}^{\infty} \frac{1}{j}(-1)^{j+1} \sin(2jx).
\end{align*}
\]
To show the validity of this we show the graph of \( f(x) = x \) (dashed line) together with a graph of the first \( k \) terms of the series of sines. For \( k \in \{1, 2, 4, 8, 16, 32\} \) this gives:

The series converges to the correct function.

**Example**

We can also use sinusoids to model functions that seem completely alien to sines, e.g., the step function

\[
f(x) = \begin{cases} 
0 & \text{if } 0 \leq x \leq 1 \\
1 & \text{if } 1 < x \leq 2 \\
2 & \text{if } 2 < x \leq 3,
\end{cases}
\]

which can be modeled by

\[
f(x) = 1 - \frac{3 \sin(\frac{2\pi x}{\pi})}{\pi} - \frac{3 \sin(\frac{4\pi x}{2\pi})}{4\pi} - \frac{3 \sin(\frac{5\pi x}{4\pi})}{5\pi} - \frac{3 \sin(\frac{10\pi x}{5\pi})}{7\pi} - \frac{3 \sin(\frac{14\pi x}{7\pi})}{11\pi} + \ldots
\]

as these graphs show for 2,4,8,16,32, and 64 terms of this series:
7.1 The relation between digital images and sinusoids

7.1.1 The Fourier transform

Given a function $f(x)$, how can we determine how often each frequency\(^1\) occurs in it? And how do these frequencies relate to $f(x)$? The answer to these questions can be found using the Fourier transform of $f(x)$, denoted by $\mathcal{F}[f](u)$:

$$
F(u) = \mathcal{F}[f](u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx,
$$

where $i$ is the complex number defined by $i^2 = -1$.

The Fourier transform $\mathcal{F}$ transforms the function $f(x)$ into a new function $F(u)$ of a new variable $u$. The value of $F(u)$ gives information on the spatial frequency $u$ in the function $f$. Note that $F(u)$ may have complex values. Typically, we are interested in ‘how much’ frequency $u$ is present in the image $f$: this can be measured by the magnitude of the Fourier transform at frequency $u$, i.e. $|F(u)|$. Another often used measure is the magnitude squared; the power spectrum $|F(u)|^2$.

Fourier transforming a function is an invertible operation; the original function can be computed from the Fourier transform by the inverse (or backward) Fourier transformation $\mathcal{F}^{-1}$:

$$
\hat{f}(x) = \mathcal{F}^{-1}[F](x) = \int_{-\infty}^{\infty} F(u) e^{i2\pi ux} du.
$$

Intermezzo

There are many different definitions of the Fourier transform and its inverse. The definitions most commonly used in the physical sciences, mathematics, and electrical engineering all differ slightly. The basic behavior of the transform is the same in all cases, though. Common differences are:

\(^1\)Where frequency is defined as the number of periods (cycles) of a sinusoid per unit length, i.e., $\sin(2\pi x)$ has frequency 1, $\sin(x)$ has frequency $\frac{1}{2\pi}$, etc.
- Swapping the minus sign in the exponent between the forward and the backward transformation.
- Omitting the factor $2\pi$ in the exponent.
- Including a global factor $\frac{1}{\sqrt{2\pi}}$.

**Example**

Explicit computation of the Fourier transform of a given analytic function $f(x)$ is often a difficult task. In this example we do not show the mathematics used to compute the transforms. If you can’t work it out by yourself, this is not important. Such mathematical skills are not necessary to understand and use the Fourier transform in practice.

Given the function

$$f(x) = \cos(2\pi x) = -1 \quad -0.5 \quad 0.5 \quad 1$$

what do we expect for the Fourier transform $F(u)$? Since $f$ contains only one frequency (frequency one), we expect $F(u)$ to have some value if $u$ equals 1, and be zero everywhere else. This is almost what happens; the Fourier transform equals

$$F(u) = \frac{1}{2}(\delta(u - 1) + \delta(u + 1)) =$$

where $\delta$ is the delta function defined in chapter 3. $F(u)$ not only has a non-zero value at the expected frequency (1), but also at the negative frequency mirror ($-1$). This phenomenon is common to the Fourier transform.²

²Without going into the mathematical details, a hint to explain the occurrence of negative frequency mirrors is the fact that the function $\cos(2\pi x)$ equals the function $\cos(-2\pi x)$. 
Example

The function

\[ f(x) = \sin(\pi x) + \sin(2\pi x) + \sin(4\pi x) = \]

is a sum of three distinct sinusoids, so we expect the Fourier transform to be zero everywhere, except at the three occurring frequencies 0.5, 1, and 2 (and their negative mirrors −0.5, −1, and −2). This is indeed what happens, since the Fourier Transform equals

\[
F(u) = -\frac{1}{\sqrt{2}} \left( \delta(u - \frac{1}{2}) + \delta(u - 1) + \delta(u - 2) + \\
-\delta(u + \frac{1}{2}) - \delta(u + 1) - \delta(u + 2) \right)
\]

Note that the graph shows the imaginary part \( I(F(u)) \) of the complex value \( F(u) \). In this example, \( F(u) \) is a pure imaginary number; its real component is zero. In general, odd functions (like the sine function) have an imaginary Fourier transform, and even functions (like the cosine in the previous example) have a real Fourier transform. In practice, signals and images are seldom so well-behaved as to be even or odd, so we may expect their Fourier transforms to be all over the complex plane.

Example

In the examples above, the Fourier transforms had a “spike” character (consisted only of delta functions), because the functions \( f(x) \) contained only distinct frequency components. In this example we choose for \( f \) a function that has no such distinct components.
Given

\[ f(x) = \pi e^{-2\pi|x|} = \]

then the Fourier transform equals

\[ \mathcal{F}(u) = \frac{1}{u^2 + 1} = \]

Since our primary interest is two-dimensional functions (images), we introduce the two-dimensional Fourier transform \( \mathcal{F}(u, v) \) of \( f(x, y) \):

\[ \mathcal{F}(u, v) = \mathcal{F}[f](u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)e^{-i2\pi(ux+vy)} \, dx \, dy, \]

and the inverse transform:

\[ f(x, y) = \mathcal{F}^{-1}[F](x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{F}(u, v)e^{i2\pi(ux+vy)} \, du \, dv. \]

The variable \( u \) in the Fourier transform now corresponds to frequencies of sinusoids oriented along the \( x \) direction. \( v \) Corresponds to frequencies in the \( y \) direction, in the following manner:

- \( \mathcal{F}(u, 0) \) is related to the frequency \( u \) of sinusoids occurring parallel to the \( x \) axis of the image.
- \( \mathcal{F}(0, v) \) is related to the frequency \( v \) of sinusoids occurring parallel to the \( y \) axis of the image.
7.1 The relation between digital images and sinusoids

- \( F(u, v) \) is related to the frequency \( \sqrt{u^2 + v^2} \) of sinusoids parallel to a line from the origin to \((u, v)\).

Below are some examples of Fourier transforms of images, clarifying these concepts. But first we need some conventions on displaying transforms of images.

**Intermezzo: display of the two-dimensional Fourier transform**

To avoid problems with displaying the complex-valued transform \( F(u, v) \) of an image \( f(x, y) \), a common approach is to display only the magnitude \( |F(u, v)| \) (the modulus value), and ignore any phase information of \( F(u, v) \). This makes sense if we are only interested in a picture showing ‘how much’ each frequency occurs in an image \( f \).

It is also common to display \( |F(u, v)| \) for negative values of \( u \) and \( v \). For this, the origin of the image is shifted to the image center.

Typically, the values of \( |F(u, v)| \) are such that direct display of these values results in dark, low-contrast images. To enhance contrast, \( \log(|F(u, v)| + 1) \) (or a similar remapping of \( |F(u, v)| \)) is often displayed instead.

Figure 7.1 shows examples of Fourier magnitude images of images containing only sinusoids. Note that

- Since the images contain only distinct frequencies, the Fourier images contain only ‘dots’, *i.e.*, have non-zero values only at specific locations.
- Since the coordinates of the dots equal the frequencies of the sinusoids occurring in the original image, higher image frequencies cause the dots to move away from the origin.
- The dots are oriented from the origin in the same direction as the frequency ‘wave’ in the original image. If we rotate the image (*e.g.*, consider the first and second rows), the Fourier transform rotates with it.
Figure 7.1  Examples of Fourier magnitude images (right column) of images containing only sinusoids (left and middle column). Axes have been added for clarity. See text for details.
Figure 7.2  Examples of Fourier magnitude images (right column) of real images (left column). The top example is of a binary image, the other images are grey-valued. See text for details.
The dots are oriented in the direction of the waves, so perpendicular to the edges occurring in the original image.

Figure 7.2 shows some examples of Fourier magnitude images of real images. In the second example, we see an oriented cluster in the Fourier transform perpendicular to the orientation of the orca, i.e., consistent with the sinusoids examples, where Fourier ‘dots’ are directed perpendicular to edges in the image.

7.1.2 Fourier series

At the beginning of section 7.1 we showed some examples of how functions could be written as a series of sinusoids of distinct frequency. Such a series is called a Fourier series of a function \( f \). Although it is closely related to the Fourier transform, the series and the transform are essentially different concepts:

- The Fourier series writes the function \( f \) as a sum of sinusoids, each with a distinct frequency. If the series converges, its value equals the value of \( f \).
- The Fourier transform transforms \( f(x) \) into a new function \( F \) of a new variable \( u \). The values of \( F(u) \) give information on the frequency components of \( f \) at frequency \( u \).

7.2 Image processing in the frequency domain

The frequency domain, \( (i.e., \) the domain of the variables \( u \) and \( v \) of the Fourier transform \( F(u, v) \) of an image \( f(x, y) \)), allows access to a new range of image processing operations directly modifying the image frequency characteristics, something which is usually much harder to achieve by spatial domain processing. Moreover, the frequency domain displays many interesting mathematical properties that cause that even image processing tasks that are not primarily related to image frequencies may be carried out advantageously in the frequency domain.

In this section, we will explore some typical frequency domain processing tasks, using the Fourier transform as defined in section 7.1.1.

A common frequency domain processing technique is the removal of frequencies within a certain range from an image. This technique

- computes the Fourier transform \( F(u, v) \) of an image \( f(x, y) \)
- computes a new transform \( F'(u, v) \) by setting specific values of \( F(u, v) \) to zero
computes the inverse Fourier transform \( f'(x, y) \) of \( F'(u, v) \), which equals \( f(x, y) \) without the frequencies that were set to zero.

**Example**

Given this image and its Fourier magnitude image:

Suppose we want to remove high frequencies from this image (i.e., create a low-pass filter): then we can set all frequency values \( F(u, v) \) to zero if they are farther than a certain distance from \((0, 0)\), and the Fourier magnitude image becomes:

If we transform this image back to the spatial domain, the resulting image is:
By removing the high frequencies, small details have vanished from the image, as is clear from these blow-ups of the original and low-pass filtered image:

If we compare the low-pass filter from the example above to the one defined by convolution in chapter 5, it will be clear that this new filter is much more flexible; we now have more complete control as to which frequencies are to be removed and retained by the filter.

Example

We can create a high-pass filter by setting all values of $F(u, v)$ at frequencies $(u, v)$ closer than a certain distance to $(0, 0)$ to zero. Using the image from the previous example, the Fourier magnitude image then looks like:

the inverse transform (high-pass filtered image) image then is:
In the examples above, the result image $f_r(x, y)$ is obtained by computing

$$f_r(x, y) = \mathcal{F}^{-1}[F(u, v)M(u, v)],$$

where $F(u, v)$ is the Fourier transform of $f(x, y)$ and $M(u, v)$ is a mask image. The mask image is an image that has value 0 at pixel locations $(u, v)$ where $F(u, v)$ is to be set to zero, and value 1 at pixel locations where $F(u, v)$ is not to be altered. Multiplying $F$ with the mask image $M$ is called masking $F$.

**Example**

In the last two examples, the mask images are respectively:

![Mask Images](image)

where black represents a zero value and white represents one.

A more sophisticated filter can be constructed if we allow more smooth transitions in the mask image $M$ than just "step" transitions from 0 to 1 and vice versa.
Example

A new low-pass filter can be constructed by choosing for the mask image $M$ a function $G_\sigma$, with

$$M(u, v) = G_\sigma(u, v) = e^{-2\pi^2\sigma^2(u^2+v^2)}.$$

Compare this mask image for a certain choice of $\sigma$ with the binary mask image used before:

![Mask Images](image1)

Applying this low-pass filter to our test image, i.e., computing

$$f_r(x, y) = \mathcal{F}^{-1}[F(u, v)G_\sigma(u, v)],$$

the resulting image (together with a blow-up) is:

![Resulting Images](image2)

By changing the parameter $\sigma$ we can let more (lower $\sigma$) or less (higher $\sigma$) frequencies pass.

By using smooth mask images, less artifacts in the result images are produced than when using a mask image that has sharp edges. Especially “ringing” artifacts around image objects (see blow-ups in the examples) may be avoided in this way.
In the previous section, we computed
\[ f_r(x, y) = \mathcal{F}^{-1}[F(u, v)M(u, v)], \]
where the function \( M \) was used to select which frequencies of the original image \( f \) end up in the ‘filtered’ result \( f_r \). One choice of \( M \) was a binary-valued function, often called a rectangle, boxcar or block filter, because of its shape when plotted (see figure 7.3). The frequency where \( M \) jumps from 1 to 0 is called the cutoff frequency.

A second choice for \( M \) was an exponential function \( G \) (see figure 7.3, which is called a Gaussian filter because \( G \) is the Fourier transform of a Gaussian function. A third often used filter shape is the Butterworth filter \( B \) (see figure 7.3), the simplest version of which equals
\[ B(u) = \left(1 + \left(\frac{u}{u_0}\right)^{2n}\right)^{-1}, \]
or in two dimensions
\[ B(u, v) = \left(\left(1 + \left(\frac{u}{u_0}\right)^{2n}\right)\left(1 + \left(\frac{v}{v_0}\right)^{2n}\right)\right)^{-1}, \]
where \( n \) is called the order of the function, and the parameters \( u_0 \) and \( v_0 \) determine the width of the filter. The value \( B(u) \) is always 1 at \( u = 0 \) (for positive \( n \)) and the value is \( \frac{1}{2} \) at \( u = u_0 \). The order \( n \) controls the steepness of the filter. Positive values of \( n \) shape the filter into a low-pass form, and negative values into a high-pass form. Figure 7.4 shows some examples of the Butterworth filter for different values of \( n \).

Many alternative filters exist besides the mentioned block, Gaussian and Butterworth filters, each with specific properties. Much used filters are the Bartlett filter (linear / triangular shape), the Welch filter (parabolic shape), the Chebyshev filter, and the Blackman, Hamming and Hanning filters (with shapes derived from cosines).
Figure 7.4 Example of the Butterworth filter with \( u_0 = 1 \). On the left, the filter for orders \( n = 1 \) (solid line), and \( n = 2 \) and \( n = 3 \) (dashed lines) can be seen. On the right, the same plots for \( n = -1 \) (solid line) and \( n = -2 \) and \( n = -3 \) (dashed lines).

7.2.2 Removing periodic noise

Many imaging devices display a characteristic artifact known as periodic noise, where the ideal images are distorted by the addition of a periodic wave pattern to the image. If we can establish the frequency of this wave pattern, we can often remove much of the artifact from the image by masking of the Fourier transform of the image.

Example

Given an image with periodic noise and its Fourier transform magnitude image:

we suspect the two bright spots in the Fourier image to be caused by the periodic noise. The orientation of the spots also corresponds with the orientation of the wave pattern in the original image. By masking the Fourier transform such that the value of \( F(u, v) \) in the area around the spots is set to zero, and transforming the result back to the spatial domain, we can remove most of the periodic noise artifact:
7.3 Properties of the Fourier transform

The Fourier transform has many special mathematical properties. A few are especially of interest in practical image processing, and will be covered in the next sections.

7.3.1 Separability

In the formula of the two-dimensional Fourier transform of a function $f(x, y)$, we can separate the two-dimensional integral into two one-dimensional parts:

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)e^{-i2\pi(ux+vy)} \, dx \, dy = \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} f(x, y)e^{-i2\pi ux} \, dx \right] e^{-i2\pi vy} \, dy.$$

We can therefore compute $F(u, v)$ by computing two one-dimensional integrals; namely first the integral between the square brackets, integrating to $x$, second the outer integral, integrating to $y$. This property is useful for computing the Fourier transform of an image, because we do not need to ‘traverse’ the entire two-dimensional image for each computation of a transform value. Instead, we can use the value of the integral along each image line (the integral between brackets), store them as intermediate results, and then compute the integral along each image column (outer integral).
7.3.2 Linearity

Given two images \(f(x, y)\) and \(g(x, y)\) and their respective Fourier transforms \(F(u, v)\) and \(G(u, v)\), then the following linearity relationship holds:

\[
\mathcal{F}[f(x, y) + g(x, y)] = F(u, v) + G(u, v),
\]

and, more general,

\[
\mathcal{F}[af(x, y) + bg(x, y)] = aF(u, v) + bG(u, v),
\]

where \(a\) and \(b\) are arbitrary constants. In words, this relationship says that the Fourier transform of a sum equals the sum of the Fourier transforms.

This property can, e.g., be useful when we know an image to be the sum of an ‘ideal’ image and a distorting image (noise), and we are able to estimate the Fourier transform of the noise image.

**Example**

Consider the image distorted with periodic noise from section 7.2.2:

We assume this image \(g\) to be the sum of an ideal image \(f\) and a noise image \(\eta\):

\[
g(x, y) = f(x, y) + \eta(x, y).
\]

We define the Fourier transforms of \(g\), \(f\), and \(\eta\) by \(G\), \(F\), and \(H\) respectively. We further assume that \(\eta\) contains only a single frequency (which seems to be the case when we observe the image \(g\)), that can be modeled by a sinusoid. Using measurements from the image background, we can estimate the orientation, frequency, and amplitude of \(\eta\). With this information, we can estimate \(H(u, v)\). Since \(G = F + H\), we can now approximate the ideal image \(f\) by subtracting \(H\) from \(G\), and then computing the inverse Fourier transform:

\[
f = \mathcal{F}^{-1}[F] = \mathcal{F}^{-1}[G - H].
\]

This results in effective removal of the periodic noise; the result looks like:
7.3 Properties of the Fourier transform

7.3.3 Convolution property

In previous chapters it will have become clear that convolution is a very important concept in image processing. Not only can image acquisition nearly always be described as a convolution; many image processing operations can also be formulated as convolutions. But, convolution is a mathematical operation of some complexity, and practical implementation may suffer from this. Fortunately, the Fourier transform has a property that greatly reduces the complexity of working with convolutions. Given functions (or images) \( f \) and \( g \) and their respective Fourier transforms \( F \) and \( G \):

\[
\mathcal{F}[f * g] = FG.
\]

This means that the difficult operation of convolution in the spatial domain is equivalent to the much simpler operation of multiplication in the frequency domain. We can therefore compute the convolution \( f * g \) using only Fourier transforms and multiplication:

\[
f * g = \mathcal{F}^{-1}[\mathcal{F}[f * g]] = \mathcal{F}^{-1}[FG],
\]

so computation of \( f * g \) can be carried out by:

- Computing the Fourier transforms \( F \) and \( G \) of \( f \) and \( g \) respectively.
- Computing the multiplication \( FG \).
- Computing the inverse Fourier transform \( \mathcal{F}^{-1}[FG] = f \ast g \).

Since efficient and fast programs to compute Fourier transforms of images exist and are widely available, using Fourier transforms to compute convolutions is by far the most common technique used.
Example

Suppose we wish to compute the convolution $g \ast f$ of an image $f$ with a Gaussian kernel $g(x, y) = \frac{1}{\sigma^2\pi e} e^{-\frac{x^2+y^2}{2\sigma^2}}$.

The Fourier transform of $g$ equals $G(u, v) = e^{-2\pi^2\sigma^2(u^2+v^2)}$. Assuming that we can compute the Fourier transform $F(u, v)$ of the image $f$ using standard software, we can compute the convolution result by evaluating $\mathcal{F}^{-1}[F(u, v)e^{-2\pi^2\sigma^2(u^2+v^2)}]$.

### 7.3.4 Derivative property

The Fourier transform can also be used to compute derivatives of functions: given a signal $f(x)$, the following relation holds:

$$\mathcal{F}[f'] = 2\pi iuf[f],$$

where $f'$ is the derivative of $f$. As with convolution, the computation of derivatives collapses to a simple multiplication in the Fourier domain. The relation can be extended to computing partial derivatives of higher-dimensional functions; given a function $f(x, y)$ and its Fourier transform $F(u, v)$:

$$\mathcal{F} \left[ \frac{\partial f}{\partial x} \right] = 2\pi iuF(u, v)$$

$$\mathcal{F} \left[ \frac{\partial f}{\partial y} \right] = 2\pi ivF(u, v)$$

$$\mathcal{F} \left[ \frac{\partial^2 f}{\partial x\partial y} \right] = -4\pi^2uvF(u, v)$$

etc.

Theoretically, these relations can be used to compute derivatives of images using only Fourier transforms and multiplications. In practice though, the results are not often very useful (we will explain this in chapter 9). A slight modification of this method however— which combines it with a well-chosen low-pass filter—suddenly makes it an effective tool for computing image derivatives. This extended method will be treated in detail in chapter 9.

### 7.3.5 Other properties

The Fourier transform has many more interesting properties than the ones listed above. Some are listed in table 7.1. Table 7.2 shows some much used Fourier transforms.
Table 7.1 Fourier transform properties. Upper case letters denote Fourier transforms of the corresponding lower case letter functions.

7.4 Inverse filtering

In chapter 3 we described image acquisition by a convolution process:

\[ g(x, y) = h(x, y) * f(x, y), \]

where \( g \) is the acquired image, \( f \) is the ‘ideal image’, and \( h \) is the point spread function (PSF) of the imaging device. Fourier transforming this expression gives us

\[ G(u, v) = H(u, v)F(u, v). \]

This means that if \( H \neq 0 \), we can reconstruct our ideal image \( f \) from \( F \):

\[ F(u, v) = \frac{G(u, v)}{H(u, v)}. \]

All we need is the PSF \( h \) (or a good estimate) of the imaging device, so we can compute \( H \), then \( F \), and finally \( f \). Reconstructing the ideal image \( f \) in this way is called inverse filtering, where \( \frac{1}{H} \) is the inverse filter of the convolution filter \( H \). As we are in fact reversing the convolution process, this process is also called deconvolution.

Are we now able to remove all the blur induced by the imaging device? Can we sharpen out-of-focus photographs? Unfortunately, even though we are sometimes able to enhance images to some extent, in most cases the answer to these questions is no. The reason for this is that we demanded that \( H \) not be zero, which is not a reasonable demand in practice. Values of \( H \) that are zero or close to zero cause the inverse filter to ‘blow up,’ disrupting the reconstruction.
### Fourier transform

<table>
<thead>
<tr>
<th>name</th>
<th>function</th>
<th>Fourier transform</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sinusoids</strong></td>
<td>( \cos(2\pi ax) )</td>
<td>( \frac{1}{2} (\delta(u - a) + \delta(u + a)) )</td>
</tr>
<tr>
<td></td>
<td>( \sin(2\pi ax) )</td>
<td>( -\frac{1}{2}i (\delta(u - a) - \delta(u + a)) )</td>
</tr>
<tr>
<td></td>
<td>( e^{i2\pi ax} )</td>
<td>( \delta(u - a) )</td>
</tr>
<tr>
<td></td>
<td>( e^{i2\pi(ax+by)} )</td>
<td>( \delta(u - a, v - b) )</td>
</tr>
<tr>
<td><strong>Gaussian</strong></td>
<td>( \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}x^2/\sigma^2} )</td>
<td>( e^{-\sigma^2 x^2} )</td>
</tr>
<tr>
<td><strong>2D Gaussian</strong></td>
<td>( \frac{1}{\sigma^2 2\pi} e^{-\frac{1}{2}x^2+y^2/\sigma^2} )</td>
<td>( e^{-2\sigma^2(u^2+v^2)} )</td>
</tr>
<tr>
<td><strong>Dirac-delta</strong></td>
<td>( \delta(x) )</td>
<td>( 1 )</td>
</tr>
<tr>
<td><strong>Grid (or Shah)</strong></td>
<td>( \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} \delta(x - i, y - j) )</td>
<td>( \text{Grid}(u, v) )</td>
</tr>
</tbody>
</table>
| **Rectangle**      | \( \begin{cases} 
1 & \text{if } |x| \leq \frac{1}{2} \text{ and } |y| \leq \frac{1}{2} \\
0 & \text{elsewhere} 
\end{cases} \) | \( \text{sinc}(u, v) \)         |

**Table 7.2** Some Fourier transforms.

The adverse effect of small values of \( H \) can be limited in a number of ways, one of which is to use the *pseudoinverse* filter \( H_p \). We compute

\[
F = GH_p,
\]

where \( H_p \) is defined by

\[
H_{p,1}(u, v) = \begin{cases} 
\frac{1}{H(u, v)} & \text{if } |H| > \varepsilon \\
0 & \text{if } |H| \leq \varepsilon,
\end{cases}
\]

where \( \varepsilon \) is a suitably small positive constant.

A more sophisticated approach is not to use a threshold \( \varepsilon \), but to let the value of the inverse filter smoothly go to zero if \( |H| \) gets very small. In this case we define the pseudoinverse filter \( H_p \) by

\[
H_{p,2}(u, v) = \frac{1}{H(u, v)} \frac{|H(u, v)|^2}{|H(u, v)|^2 + S(u, v)}.
\]

If \( S(u, v) \) is determined optimally from the amount of noise\(^3\) at the frequencies \((u, v)\), then this filter is called a *Wiener* filter. Since the mathematics behind a Wiener filter are fairly complex, we will not detail the computation of \( S \) here. In practice, \( S \) is often set to an empirically established constant value. This usually produces acceptable results if the original image does not contain too much noise.

\(^3\)Or, more accurately, from the *signal to noise* ratio.
Example

Suppose we have an input image $f_b$ we know to be blurred by Gaussian convolution: $f_b = g * f_i$, where $g$ is a Gaussian kernel, and $f_i$ the ‘ideal’ image. Fourier transforming gives us $F_b = GF_i$, so we can theoretically find $f_i$ by computing $F^{-1}[F_b G]$. In the case where the convolution is exactly Gaussian and we know the width of the Gaussian, inverse filtering works very well. Here are $f_b$ and the reconstructed $f_i$ in this case:

![Blurred and reconstructed images](image1)

But look what happens to the reconstructed $f_i$ if we estimate the width of the Gaussian incorrectly; in these cases a 3\% and 17\% error in the width $\sigma$:

![Reconstructed images with errors](image2)

Now, small values of $G$ have caused severe artifacts. Using the pseudoinverse filter $H_{p,1}$ in the latter case can remove the worst artifacts (using a suitable value for $\varepsilon$):
Using $H_{p,2}$ with a suitable constant value for $S$ gives us a somewhat better result:

We can use inverse and pseudoinverse filtering to reverse—or, at least, attempt to reverse—any convolution process. An example of this is the removal of motion blur. Blurring of images caused by movement of the imaging device or the object imaged is a very common problem in many imaging areas, notably those where imaging requires a relatively long exposure time.

**Example**

Suppose we have the following image:
where we suppose this image to be blurred by a movement of the camera during acquisition. We assume that in the blurred image, each pixel value is the average of the values of a number of pixels in the ideal image that are along a line in the direction of blur. So the blurring can be described as a convolution with a rectangular kernel, such as the ones below:

where the height axis shows the kernel value, and the other two axis span the image plane, so the convolution effect will be blurring in a horizontal, diagonal, and vertical direction respectively. From the input image, we need to estimate the direction of blur, which determines the orientation of the kernel, and the extent of the blurring, which determines the width of the kernel. In this particular case, we judge the blurring direction to be vertical, and gauged the amount of blur by trial-and-error. The blurring kernel $h$ is a vertically oriented rectangular filter, which we model by $h(x, y) = \text{rect}_c(y)\delta(x)$, where $\text{rect}_c(y)$ is a rectangular kernel with value $\frac{1}{2c}$ if $|y| \leq c$ and zero value otherwise. The delta function $\delta(x)$ ensures the kernel is infinitely narrow and has values only along the $y$-axis. Although this may look like a function that is difficult to work with, its Fourier transform is quite friendly: $H(u, v) = \frac{\sin(2\pi cv)}{2\pi cv}$. Using the pseudoinverse filter $H_{p,2}$ with a suitable constant value for $S$, we are able to deblur the input image to:

---

### 7.5 The Discrete Fourier Transform

In the previous sections we have used Fourier transforms of images, even though we have formally defined the Fourier transform for continuous functions only. Since a dig-
ital image is not a continuous function, but a matrix of sampled data, we have used an adapted type of Fourier transform, called the discrete Fourier transform (DFT). The DFT $F(u, v)$ of an image $f(x, y)$ with dimensions $d_x \times d_y$ is defined by

$$F(u, v) = \sum_{x=0}^{d_x-1} \sum_{y=0}^{d_y-1} f(x, y)e^{-i2\pi\left(\frac{ux}{d_x} + \frac{vy}{d_y}\right)},$$

where $u \in \{0, 1, \ldots, d_x - 1\}$, and $v \in \{0, 1, \ldots, d_y - 1\}$. The inverse transformation is defined by

$$f(x, y) = \frac{1}{d_x d_y} \sum_{u=0}^{d_x-1} \sum_{v=0}^{d_y-1} F(u, v)e^{i2\pi\left(\frac{ux}{d_x} + \frac{vy}{d_y}\right)}.$$

The DFT image of an $d_x \times d_y$ image is again a $d_x \times d_y$ image, which generally is complex-valued. Computing a larger DFT image is not useful, because the $d_x \times d_y$ image already captures all of the frequency information up until the Nyquist frequency of the original.

The DFT has all of the properties we defined for the ordinary Fourier transform. Many algorithms can be found that efficiently compute the DFT of an image. The algorithm complexity can be reduced by using the point symmetry\(^4\) and the separability of the transform, and by using smart reordering of the image data. The most common method of computing the DFT is known as the fast Fourier transform or FFT.

### 7.6 Other transforms

The Fourier transform is based on the fact that functions and images can be viewed as a sum of sinusoids. The frequency of these sinusoids is closely related to image aspects such as the level of detail in an image. The Fourier transform $F(u)$ of a function $f(x)$ gives us information on the sinusoids at frequency $u$, and the previous sections have hopefully shown that the Fourier transform is a very powerful transform in the sense that careful manipulation of $F(u)$ can be used to perform a wide range of image processing tasks.

Since the Fourier transform decomposes an image into sinusoids, these functions are called the basis functions of the Fourier transform. The fact that an image can be viewed as a sum of sinusoids raises the question whether there are alternative choices for basis functions.

\(^4\)To be more precise, the transform is conjugate symmetric if the input image has real values.
functions, and hence if there are alternative transforms that may be useful for image processing. The answer is yes: many families of functions are suitable as basis functions; only basic mathematical requirements need to be fulfilled. The associated transforms can have very diverse properties. Some transforms may have properties similar to the Fourier transform, yet may be easier to compute. Others may be better suited to compactly represent the shapes and sizes of objects. Still others may present the essential image content compactly, and so are useful in image compression. Some dozen or so transforms pop up regularly in image processing applications, but none have gained the popularity (or, in fact, the usefulness) of the Fourier transform. Well-known transforms are, e.g., the Hartley, cosine, sine, Hadamard, Karhunen-Loéve, Slant, and various Wavelet transforms. A few of these transforms are listed below.

The Hartley transform

If we rewrite the Fourier transform $F(u)$ of $f(x)$ in terms of sines and cosines:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx$$

$$= \int_{-\infty}^{\infty} f(x) (\cos(2\pi ux) - i \sin(2\pi ux)) dx,$$

then the Hartley transform $H(u)$ is defined by leaving out the complex $i$:

$$H(u) = \int_{-\infty}^{\infty} f(x) (\cos(2\pi ux) - \sin(2\pi ux)) dx,$$

and the discrete Hartley transform (DHT) is defined by

$$H(u) = \sum_{x=0}^{d-1} f(x) \left( \cos \frac{2\pi ux}{d} - \sin \frac{2\pi ux}{d} \right).$$

A fast Hartley transform (FHT) can also be defined in a similar fashion as with the Fourier transform.

The Hartley transform is a computationally attractive alternative for the Fourier transform. Note that the Hartley transform produces real-valued output when a real-valued input image is used, while the Fourier transform is generally complex-valued in this case. On the downside, expressions and properties tend to be more complicated than in the Fourier case. It is possible to use the Hartley transform in all of the examples in the
sections on the Fourier transform, producing the same results, but –as mentioned– the expressions are more difficult and somewhat less easy to work with.

**The cosine transform**

The discrete cosine transform (DCT) $F(u)$ of a function $f(x)$ is defined by

$$F(u) = c(u) \sum_{x=0}^{d-1} f(x) \cos \frac{\pi u (2x + 1)}{2d},$$

where $c(u)$ is defined by

$$c(u) = \begin{cases} \sqrt{d-1} & \text{if } u = 0 \\ \sqrt{2} \sqrt{d-1} & \text{otherwise} \end{cases}$$

The cosine transform can be computed much faster than the Fourier transform. Its most used property is the fact that essential image content can be represented in a very compact form. Hence the cosine transform is much used in image compression techniques.

**The wavelet transform**

The wavelet transform is a generalized type of transform in the sense that its basis functions are not pre-determined. In the transforms treated in the above, the basis functions were a particular combination of sines and cosines in each case. With the wavelet transform the basis functions are all computed from a single *mother wavelet* $\psi(x)$, which can be any function as long as it obeys certain mathematical rules. The other basis functions are created by scaling and translating the mother wavelet.

Given a mother wavelet $\psi(x)$, the family of basis functions is generated by scaling with a factor $a$ and translation by a factor $b$:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{|a|}} \psi \left( \frac{x - b}{a} \right).$$

The continuous wavelet transform is then defined by

$$F(a, b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} f(x) \psi \left( \frac{x - b}{a} \right) dx.$$ 

The discrete form is somewhat more complex than may be expected, because the possible values of $a$ and $b$ are restricted by the use of a discrete grid. A common definition for integer values of $a$ and $b$ is

$$F(a, b) = \int_{-\infty}^{\infty} f(x) 2^{-\frac{a}{2}} \psi(2^{-a} x - b) dx.$$
A popular choice for the mother wavelet is the *Haar function* $\varphi(x)$, which is defined by

$$
\varphi(x) = \begin{cases} 
1 & \text{if } 0 \leq x < \frac{1}{2} \\
-1 & \text{if } \frac{1}{2} \leq x < 1 \\
0 & \text{otherwise}
\end{cases}
$$

The blocky shape of this function makes it especially suitable for use with digital images.

Being a general transform, the applications of the wavelet transform are extensive and diverse. Since it is a fairly new transform, its scope is not well known yet, but it has already proven its worth in *multiresolution* image analysis, which will be treated in chapter 9.

**The Karhunen-Loève transform**

This transform is also known as *principal component analysis*. In the social sciences, it is called *factor analysis*. In yet other areas of science, it appears under other names such as the Hotelling transform or eigenvalue analysis.

The Karhunen-Loève transform (KLT) is different from other transforms in the sense that its basis functions are determined from the image or function itself. In general, this transform is especially useful for getting rid of redundant parameters in problems that have many parameters. For example: suppose a scientist wishes to study the occurrence of a certain phenomenon, and has compiled a list of twenty parameters that may be connected to the occurrence of this phenomenon. (One may think of, e.g., a heart attack as the phenomenon, and blood pressure, heart rate, etc. as parameters.) It may be that some of the parameters in his list are redundant. It may also be that only a combination of certain parameters is significant to the occurrence of the phenomenon. The KLT is useful in determining whether these are the case.

Let us consider the two parameter example from figure 7.5, where the two parameters are represented by $x$ and $y$. In this figure, each closed dot represents the $x$ and $y$ parameter values at the occurrence of the phenomenon. An open dot represents a measurement of $x$ and $y$ when no phenomenon occurred. By observing the picture, it will be obvious that the parameters $u$ and $v$ are much better suited for discriminating between the occurrence and non-occurrence of the phenomenon; just take a look at the $(x, y)$ and the $(u, v)$ coordinates of each dot. The $u$ and $v$ axes are called the *principal axes* of the cloud of black dots. The KLT is used for transforming from $(x, y)$ to $(u, v)$ space. We will not cover all of the necessary mathematics of the KLT here, but an equivalent technique is covered in chapter 8.
Figure 7.5 The principal axes $u$ and $v$ of a data cloud.

The cloud of black dots can be viewed as a grey-valued image $f(x, y)$ if we say that the grey value $f(x, y)$ equals the number of times a black dot occurs in pixel $(x, y)$. By this simile, we can use the KLT to find the center and orientation of roughly elliptical shapes in grey-valued images.