7. Linear Filtering

7.1 Convolution

The one-dimensional convolution sum, Equation (2.5), formed the basis for much of our discussion on discrete-time signals and systems. Similarly the two-dimensional convolution sum will form the basis from which we begin our discussion on image processing and computer vision.

The 1-D convolution sum extends naturally to higher dimensions. Consider an image $f[x, y]$ and a two-dimensional filter $h[x, y]$. The 2-D convolution sum is then given by:

$$g[x,y] \;=\; \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} f[k,l]h[x-k,y-l]. \tag{7.1}$$

In 1-D, the intuition for a convolution is that of computing inner products between the filter and signal as the filter "slides" across the signal. The same intuition holds in 2-D. Inner products are computed between the 2-D filter and underlying image as the filter slides from left-to-right/top-to-bottom.

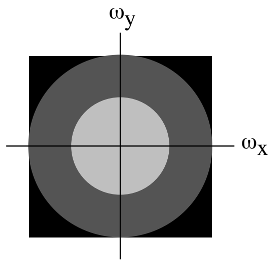

Figure 7.1 2-D Frequency

In the Fourier domain, this convolution is equivalent to multiplying the, now 2-D, Fourier transforms of the filter and image, where the 2-D Fourier transform is given by:

$$F[\omega_x,\omega_y] \;=\; \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} f[k,l]e^{-i(\omega_x k+\omega_y l)}. \tag{7.2}$$

The notion of low-pass, band-pass, and high-pass filtering extends naturally to two-dimensional images. Shown in Figure 7.1 is a simplified decomposition of the 2-D Fourier domain parameterized by ω_x and $\omega_y \in [-\pi, \pi]$. The inner disc corresponds to the lowest frequencies, the center annulus to the middle (band) frequencies, and the outer dark area to the highest frequencies.

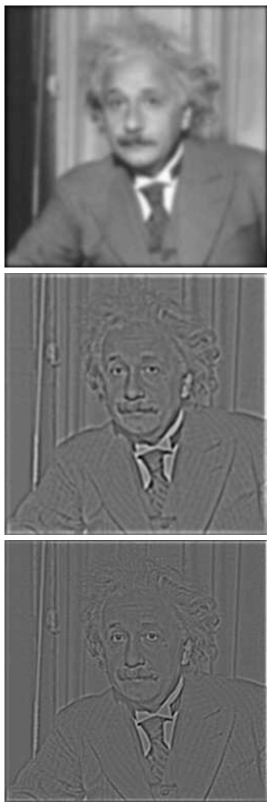

Figure 7.2 Low-, Band-, High-pass

Two of the most common (and opposing) linear filtering operations are *blurring* and *sharpening*. Both of these operations can be accomplished with a 2-D filter and 2-D convolution, or more efficiently with a 1-D filter and a pair of 1-D horizontal and vertical convolutions. For example, a 2-D convolution with the blur filter:

$$\begin{pmatrix} 0.0625 & 0.1250 & 0.0625 \\ 0.1250 & 0.2500 & 0.1250 \\ 0.0625 & 0.1250 & 0.0625 \end{pmatrix}$$

can be realized by convolving in the horizontal and vertical directions with the 1-D filter:

$$\text{blur} \; = \; \begin{pmatrix} 0.25 & 0.50 & 0.25 \end{pmatrix}. \qquad (7.3)$$

That is, an outer-product of the 1-D filter with itself yields the 2-D filter - the filters are *xy-separable*. The separability of 2-D filters is attractive for two reasons: (1) it is computationally more efficient and (2) it simplifies the filter design. A generic blur filter may be constructed from any row of the binomial coefficients:

$$\begin{array}{ccccccccccccc}
 & & & & & 1 & & 1 & & & & & \\
 & & & & 1 & & 2 & & 1 & & & & \\
 & & & 1 & & 3 & & 3 & & 1 & & & \\
 & & 1 & & 4 & & 6 & & 4 & & 1 & & \\
 & 1 & & 5 & & 10 & & 10 & & 5 & & 1 & \\
1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1
\end{array}$$

where each row (filter) should be normalized by it's sum (i.e., blur filters should always be unit-sum so as not to increase or decrease the mean image intensity). The amount of blur is then directly proportional to the size of the filter. Blurring simply reduces the high-frequency content in an image. The opposing operation, sharpening, is meant to enhance the high-frequencies. A generic separable sharpening filter is of the form:

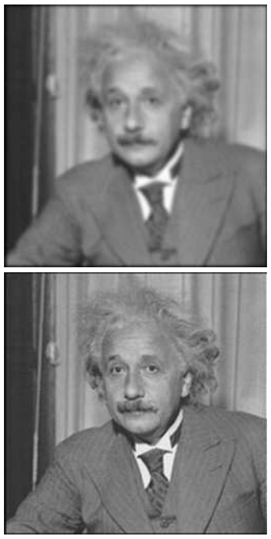
Figure 7.3 Blur and Sharpen

$$\text{sharp} \; = \; \begin{pmatrix} 0.08 & -1.00 & 0.08 \end{pmatrix}. \qquad (7.4)$$

This filter leaves the low-frequencies intact while enhancing the contribution of the high-frequencies. Shown in Figure 7.3 are results from blurring and sharpening.

7.2 Derivative Filters

Discrete differentiation forms the foundation for many applications in image processing and computer vision. We are all familiar with the definition of the derivative of a *continuous* signal $f(x)$:

$$D\{f(x)\} \; = \; \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}. \qquad (7.5)$$

This definition requires that the signal $f(x)$ be well defined for all $x \in \mathcal{R}$. So, does it make sense to differentiate a *discretely* sampled signal, $D\{f[x]\}$, which is only defined over an integer sampling lattice? Strictly speaking, no. But our intuition may be that this is not such an unreasonable request. After all, we know how to differentiate $f(x)$, from which the sampled signal $f[x]$ was derived, so why not just differentiate the continuous signal $f(x)$ and then sample the result? Surely this is what we have in mind when we ask for the derivative of a sampled signal. But one should not be fooled by the seeming simplicity of our intuition, as we will soon discover the design of an accurate and efficient discrete derivative operator will prove to be sufficiently challenging.

Recall from earlier chapters that under certain conditions (Nyquist theory), the relationship between the continuous and sampled signals can be expressed precisely as:

$$f(x) = f[x] \star h(x), \tag{7.6}$$

where $h(x) = \sin(\pi x/T)/(\pi x/T)$ is the ideal sync, and $\star$ is the convolution operator. Now, applying the continuous differential operator to both sides yields:

$$D\{f(x)\} = D\{f[x] \star h(x)\}, \tag{7.7}$$

and expressing the right-hand side in terms of the convolution sum:

$$\begin{aligned} D\{f(x)\} &= D\left\{ \sum_{k=-\infty}^{\infty} f[k]h(x-k) \right\} \\ &= \sum_{k=-\infty}^{\infty} f[k]D\{h(x-k)\} \\ &= f[x] \star D\{h(x)\}. \end{aligned} \tag{7.8}$$

Notice that the derivative operator $D\{\cdot\}$ is being applied only to continuous entities. Having computed the desired derivatives, we need only sample the results, denoting $S\{\cdot\}$ as the sampling operator:

$$\begin{aligned} S\{\ D\{f(x)\}\ \} &= f[x] \star S\{\ D\{h(x)\}\ \} \\ S\{f'(x)\} &= f[x] \star S\{h'(x)\} \\ f'[x] &= f[x] \star h'[x]. \end{aligned} \tag{7.9}$$

On the left-hand side of the above equation is the desired quantity, the derivative of the sampled signal. On the right-hand side is a discrete convolution between two known quantities, the sampled derivative of the sync and the original sampled signal. The derivative of the sync can be expressed analytically by simply differentiating the sync function:

$$h'(x) = \frac{\pi^2 x/T^2 \cos(\pi x/T) - \pi/T \sin(\pi x/T)}{(\pi x/T)^2}, \tag{7.10}$$

where T is the sampling period at which $f(x)$ was sampled. So, if the signal $f(x)$ is sampled above the Nyquist rate and if it is in

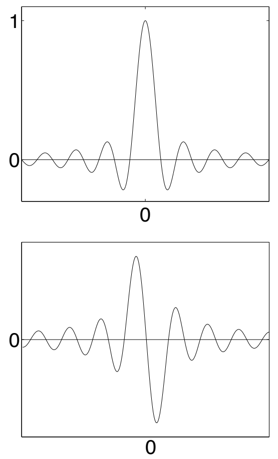

Figure 7.4 Ideal sync and its derivative

fact differentiable, then Equation (7.9) tells us that we can exactly compute the derivative of the sampled signal $f[x]$, an altogether happy ending.

If you are feeling a bit uneasy it is for a good reason. Although mathematically correct, we have a solution for differentiating a discretely sampled signal that is physically unrealizable. In particular the derivative of the sync, $h'(x)$, is spatially infinite in extent, meaning that it cannot be implemented on a finite machine. And even worse, $h'(x)$ falls off slowly from the origin so that truncation will cause significant inaccuracies. So we are going to have to part with mathematical perfection and design a finite-length filter.

To begin we need to compute the frequency response of the ideal derivative filter. We can compute the response indirectly by first expressing $f[x]$ in terms of its Fourier series:

$$f[x] = \frac{1}{2\pi} \sum_{\omega=-\pi}^{\pi} F[\omega] e^{i\omega x}, \tag{7.11}$$

and then differentiating both sides with respect to x:

$$\begin{aligned} D\{f[x]\} &= \frac{1}{2\pi} \sum_{\omega=-\pi}^{\pi} F[\omega] D\{e^{i\omega x}\} \\ &= \frac{1}{2\pi} \sum_{\omega=-\pi}^{\pi} i\omega F[\omega] e^{i\omega x}. \end{aligned} \tag{7.12}$$

Differentiation in the space domain is then seen to be equivalent to multiplying the Fourier transform $F[\omega]$ by an imaginary ramp $i\omega$. And since multiplication in the frequency domain is equivalent to convolution in the space domain, an imaginary ramp is the frequency response of the ideal derivative filter. Trying to directly design a finite length filter to this response is futile because of the discontinuity at $-\pi/\pi$, which of course accounts for the spatially infinite extent of $h'(x)$. So we are resigned to designing a filter with a *periodic* frequency response that "best" approximates a ramp. The simplest such approximation is that of a sinusoid where, at least in the low-frequency range, the match is reasonably good (i.e., $\sin(\omega) = \omega$, for small ω). Employing the least-squares filter design technique (Equation (4.8)) we formulate a quadratic error function to be minimized:

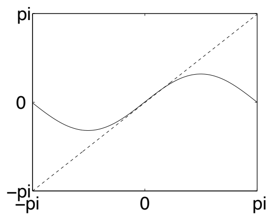

Figure 7.5 Ideal and approximate derivative frequency response

$$E(\vec{h}) = |\, M\vec{h} - \vec{H}\, |^2, \tag{7.13}$$

where M is the $N \times n$ Fourier matrix (Equation (2.28)), $\vec{H}$ is the N sampled desired frequency response, and n the filter size is.

To minimize we differentiate, set equal to zero and solve for the minimal solution:

$$\vec{h} = (M^t M)^{-1} M^t \vec{H} \tag{7.14}$$

Since the desired frequency response, a sinusoid, has only two degrees of freedom, amplitude and phase, a 2-tap filter will suffice (i.e., $n = 2$). The resulting filter is of the form $\vec{h} = (\, 0.5 \quad -0.5 \,)$. Intuitively this is exactly what we should have expected - for example, applying this filter via a convolution and evaluating at, arbitrarily, $n = 0$ yields:

$$\begin{aligned} f'[x] &= h[x] \star f[x] \\ &= \sum_{k=-\infty}^{\infty} h[x-k] f[k] \\ f'[0] &= h[1]f[-1] + h[0]f[0] \\ &= 0.5f[0] - 0.5f[-1]. \end{aligned} \tag{7.15}$$

Note that the derivative is being approximated with a simple two-point difference, that is, a discrete approximation to the continuous definition in Equation (7.5). We could of course greatly improve on this filter design. But since we are really interested in multi-dimensional differentiation, let's put aside further analysis of the one-dimensional case and move on to the two-dimensional case.

It has been the tendency to blindly extend the one-dimensional design to higher-dimensions, but, as we will see shortly, in higher-dimensions the story becomes slightly more complicated. In the context of higher-dimensional signals we first need to consider partial derivatives. For example the partial derivative of a two dimensional signal $f(x, y)$ in it's first argument is defined as:

$$\begin{aligned} f_x(x,y) &\equiv \frac{\partial f(x,y)}{\partial x} \\ &= \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}. \end{aligned} \tag{7.16}$$

According to the Nyquist theory, the continuous and discrete signals (if properly sampled) are related by the following equality:

$$f(x,y) = f[x,y] \star h(x,y), \tag{7.17}$$

where $h(x,y) = \frac{\sin(\pi x/T)\sin(\pi y/T)}{\pi^2 xy/T^2}$ is the two-dimensional ideal sync. As before we apply the continuous partial differential operator to both sides:

$$D_x\{f(x,y)\} = f[x,y] \star D_x\{h(x,y)\}, \tag{7.18}$$

noting again that the differential operator is only applied to continuous entities. Since the two-dimensional sync is separable (i.e., $h(x,y) = h(x) \star h(y)$), the above equation can be rewritten as:

$$\begin{aligned} f_x(x,y) &= f[x,y] \star D_x\{h(x) \star h(y)\} \\ &= f[x,y] \star D_x\{h(x)\} \star h(y) \\ &= f[x,y] \star h'(x) \star h(y). \end{aligned} \tag{7.19}$$

And finally, sampling both sides gives an expression for the partial derivative of the discretely sampled two-dimensional signal:

$$\begin{aligned} S\{f_x(x,y)\} &= f[x,y] \star S\{h'(x)\} \star S\{h(y)\} \\ f_x[x,y] &= f[x,y] \star h'[x] \star h[y]. \end{aligned} \tag{7.20}$$

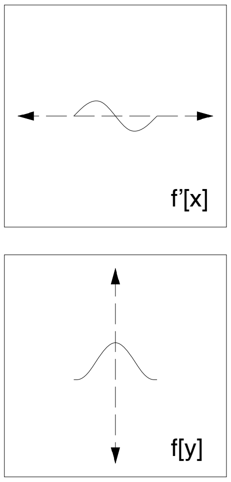

Figure 7.6 Horizontal partial differentiation

Notice that calculating the partial derivative requires a pair of one-dimensional convolutions: a derivative filter, $h'[x]$, in the dimension of differentiation, and an interpolation filter, $h[y]$, in the other dimension (for multi-dimensional signals, *all* remaining dimensions would be convolved with the interpolation filter). Since two-dimensional differentiation reduces to a pair of one-dimensional convolutions it is tempting to simply employ the same differentiation filter used in the one-dimensional case. But since a pair of filters are now required perhaps we should give this some additional thought.

In some ways the choice of filters seems trivial: chose an interpolation function $h(x)$, differentiate it to get the derivative function $h'(x)$, and sample these functions to get the final digital filters $h[x]$ and $h'[x]$. So how is this different from the one-dimensional case? In the one-dimensional case only the derivative filter is employed, whereas in the two-dimensional case we require the pair of filters. And by our formulation we know that the pair of filters should satisfy the relationship that one is the derivative of the other $h'(x) = D(h(x))$. And in fact this constraint is automatically enforced by the very nature in which the *continuous* functions are chosen, but in the final step, these functions are sampled to produce *discrete* filters. This sampling step typically destroys the required derivative relationship, and, although a seemingly subtle point, has dramatic effects on the accuracy of the resulting derivative operator. For example consider the often used Sobel derivative filters with $h[x] = (\,1 \quad \sqrt{2} \quad 1\,)/(2+\sqrt{2})$ and $h'[x] = (\,1 \quad 0 \quad -1\,)/3$. Shown in Figure 7.7 are the magnitudes of the Fourier transform of the derivative filter (solid line) and the interpolation filter times $i\omega$ (i.e., frequency domain differentiation). If the filters obeyed the required derivative relationship than these curves would be exactly matched, which they clearly

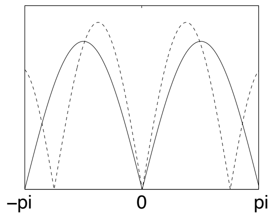

Figure 7.7
Sobel frequency response

are not. The mismatching of the filters results in gross inaccuracies in derivative measurements. Let's see then if we can design a better set of filters.

We begin by writing down the desired relationship between the derivative and interpolation filters, most conveniently expressed in the frequency domain:

$$H'(\omega) = i\omega H(\omega), \tag{7.21}$$

from which we can write a weighted least-squares error functional to be minimized:

$$E(H, H') = \int d\omega \; \left[W(\omega)(i\omega H(\omega) - H'(\omega))\right]^2, \tag{7.22}$$

where $W(\omega)$ is a frequency weighting function. Next, we write a discrete approximation of this continuous error functional over the n-vectors $\vec{h}$ and $\vec{h'}$ containing the sampled derivative and interpolation filters, respectively:

$$E(\vec{h}, \vec{h'}) = |W(F'\vec{h} - F\vec{h'})|^2, \tag{7.23}$$

where the columns of the matrix $F_{m\times n}$ contain the first n Fourier basis functions (i.e., a discrete-time Fourier transform), the matrix $F'_{m\times n} = i\omega F_{m\times n}$, and $W_{m\times m}$ is a diagonal frequency weighting matrix. Note that the dimension n is determined by the filter size and the dimension m is the sampling rate of the continuous Fourier basis functions, which should be chosen to be sufficiently large to avoid sampling artifacts. This error function can be expressed more concisely as:

$$E(\vec{u}) = |M\vec{u}|^2, \tag{7.24}$$

where the matrix M and the vector $\vec{u}$ are constructed by "packing together" matrices and vectors:

$$M = \left(WF' \quad | \quad -WF \right) \qquad \text{and} \qquad \vec{u} = \begin{pmatrix} \vec{h} \\ \vec{h'} \end{pmatrix}. \tag{7.25}$$

The minimal unit vector $\vec{u}$ is then simply the minimal-eigenvalue eigenvector of the matrix M^tM. After solving for $\vec{u}$, the derivative and interpolation filters can be "unpacked" and normalized so that the interpolation filter is unit sum. Below are the resulting filter values for a 3-tap and 5-tap filter pair.

Shown in Figure 7.8 are the matching of these filters in the frequency domain. Notice that the 5-tap filters are nearly perfectly matched.

h	0.223755	0.552490	0.223755
h'	0.453014	0.0	-0.453014

h	0.036420	0.248972	0.429217	0.248972	0.036420
h'	0.108415	0.280353	0.0	-0.280353	-0.108415

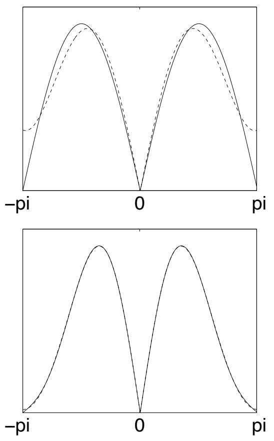

Figure 7.8
Frequency response of matched derivative filters

Higher-order derivative filters can be designed by replacing the initial constraint in Equation (7.21) with $H'(\omega) = (i\omega)^k H(\omega)$ for a k^{th} order derivative.

A peculiar aspect of this filter design is that nowhere did we explicitly try to model a specified frequency response. Rather, the design fell naturally from the relationship between the continuous- and discrete-time signals and the application of the continuous derivative operator, and in this way is quite distinct from the one-dimensional case. The proper choice of derivative filters can have a dramatic impact on the applications which utilize them. For example, a common application of differential measurements is in measuring motion from a movie sequence $f(x, y, t)$. The standard formulation for motion estimation is:

$$f_x(x, y, t)v_x(x, y) + f_y(x, y, t)v_y(x, y) + f_t(x, y, t) = 0, \quad (7.26)$$

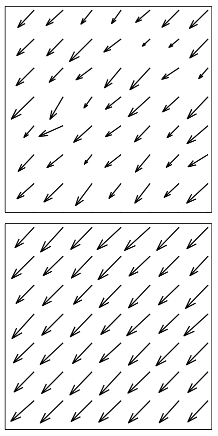

Figure 7.9
Differential motion estimation

where the motion vector is $\vec{v} = (\, v_x \quad v_y \,)^t$, and $f_x(\cdot)$, $f_y(\cdot)$, and $f_t(\cdot)$ are the partial derivatives with respect to space and time. Shown in Figure 7.9 are the resulting motion fields for a simple translational motion with the Sobel (top panel) and matched (bottom) derivative filters used to compute the various derivatives. Although these filters are the same size, the difference in accuracy is significant.

7.3 Steerable Filters

In the previous section we showed how to compute horizontal and vertical partial derivatives of images. One may naturally wonder how to compute a derivative in an arbitrary direction. Quite remarkably it turns out that we need not design a new set of filters for each possible direction because the derivative in any direction can be synthesized from a linear combination of the horizontal and vertical derivatives. This property of derivatives has been termed *steerability*. There are several formulations of this property, we chose to work in the frequency domain where differentiation takes on a particularly simple form.

To begin, we express a two-dimensional image with respect to its Fourier series:

$$f(x,y) = \sum_{\omega_x=-\pi}^{\pi} \sum_{\omega_y=-\pi}^{\pi} F(\omega_x,\omega_y) e^{-j(\omega_x x + \omega_y y)}. \tag{7.27}$$

Differentiating [8] both sides with respect to x gives:

$$f_x(x,y) = \sum_{\omega_x=-\pi}^{\pi} \sum_{\omega_y=-\pi}^{\pi} -j\omega_x F(\omega_x,\omega_y) e^{-j(\omega_x x + \omega_y y)}. \tag{7.28}$$

That is, in the frequency domain differentiation in the horizontal direction u is equivalent to multiplying the Fourier transform $F(\omega_x,\omega_y)$ by an imaginary horizontal ramp, $-j\omega_x$. Similarly, the vertical partial derivative in v is:

$$f_y(x,y) = \sum_{\omega_x=-\pi}^{\pi} \sum_{\omega_y=-\pi}^{\pi} -j\omega_y F(\omega_x,\omega_y) e^{-j(\omega_x x + \omega_y y)}. \tag{7.29}$$

Now, differentiation in the vertical direction v is equivalent to multiplying the Fourier transform by an imaginary vertical ramp. This trend generalizes to arbitrary directions, that is, the partial derivative in any direction α can be computed by multiplying the Fourier transform by an imaginary oriented ramp $-j\omega_\alpha$:

$$f_\alpha(x,y) = \sum_{\omega_x=-\pi}^{\pi} \sum_{\omega_y=-\pi}^{\pi} -j\omega_\alpha F(\omega_x,\omega_y) e^{-j(\omega_x x + \omega_y y)}, \tag{7.30}$$

where the oriented ramp can be expressed in terms of the horizontal and vertical ramps:

$$\omega_\alpha = \cos(\alpha)\omega_x + \sin(\alpha)\omega_y. \tag{7.31}$$

Substituting this definition back into the partial derivative in α, Equation (7.30), gives:

$$\begin{aligned} f_\alpha(x,y) &= \sum_{\omega_x=-\pi}^{\pi} \sum_{\omega_y=-\pi}^{\pi} -j[\cos(\alpha)\omega_x + \sin(\alpha)\omega_y] F(\omega_x,\omega_y) e^{-j(\omega_x x + \omega_y y)} \\ &= \cos(\alpha) \sum_{\omega_x=-\pi}^{\pi} \sum_{\omega_y=-\pi}^{\pi} -j\omega_x F(\omega_x,\omega_y) e^{-j(\omega_x x + \omega_y y)} \\ &\quad + \sin(\alpha) \sum_{\omega_x=-\pi}^{\pi} \sum_{\omega_y=-\pi}^{\pi} -j\omega_y F(\omega_x,\omega_y) e^{-j(\omega_x x + \omega_y y)} \\ &= \cos(\alpha) f_x(x,y) + \sin(\alpha) f_y(x,y). \end{aligned} \tag{7.32}$$

[8]Recall that the derivative of an exponential is an exponential, so that according to the chain rule, $D_x\{e^{ax}\} = ae^{ax}$.

Notice that we obtain the horizontal and vertical derivatives when $\alpha = 0$ and $\alpha = 90$. This equation embodies the principle of steerability - the derivative in *any* direction α can be synthesized from a linear combination of the partial horizontal and vertical derivatives, $f_x(x,y)$ and $f_y(x,y)$. Pause to appreciate how remarkable this is, a pair of directional derivatives is sufficient to represent an infinite number of other directional derivatives, i.e., α can take on any real-valued number.

From the previous section we know how to compute the horizontal and vertical derivatives via convolutions with an interpolation and derivative filter. To compute any other directional derivative no more convolutions are required, simply take the appropriate linear combinations of the horizontal and vertical derivatives as specified in Equation (7.32). Shown in Figure 7.10 from top to bottom is a disc $f(x,y)$, its horizontal derivative $f_x(x,y)$, its vertical derivative $f_y(x,y)$, and its steered derivative $f_{45}(x,y)$, where the steered derivative was synthesized from the appropriate linear combinations of the horizontal and vertical derivatives. The obvious benefit of steerability is that the derivative in any direction can be synthesized with minimal computational costs.

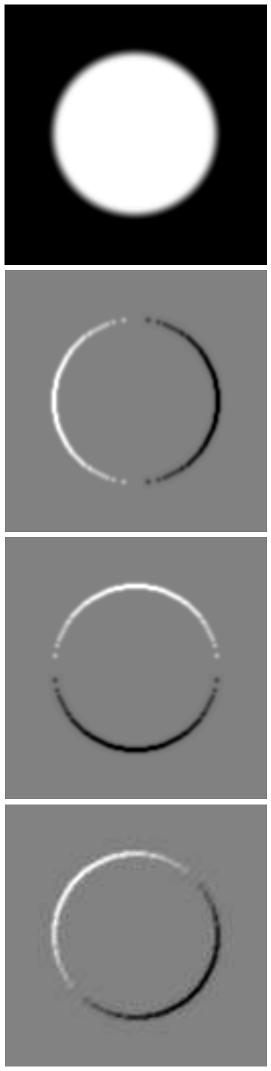
Figure 7.10 Steerability

Steerability is not limited to first-order derivatives. Higher-order derivatives are also steerable; the N^{th}-order derivative is steerable with a basis set of size $N + 1$. For example, the second-order derivative in an arbitrary direction can be synthesized as follows:

$$f_{\alpha\alpha} = \cos^2(\alpha) f_{xx} + 2\cos(\alpha)\sin(\alpha) f_{xy} + \sin^2(\alpha) f_{yy}, \qquad (7.33)$$

where for notational simplicity the spatial arguments (x,y) have been dropped, and the multiple subscripts denote higher-order differentiation. Note that three partial derivatives are now needed to steer the second-order derivative. Similarly, the third-order derivative can be steered with a basis of size four:

$$\begin{aligned} f_{\alpha\alpha\alpha} &= \cos^3(\alpha) f_{xxx} + 3\cos^2(\alpha)\sin(\alpha) f_{xxy} \\ &\qquad + 3\cos(\alpha)\sin^2(\alpha) f_{xyy} + \sin^3(\alpha) f_{yyy}. \end{aligned} \qquad (7.34)$$

You may have noticed that the coefficients needed to steer the basis set look familiar, they are the binomial coefficients that come from a polynomial expansion. More specifically, as in Equation(7.30) the N^{th}-order derivative in the frequency domain is computed by multiplying the Fourier transform by an imaginary oriented ramp raised to the N^{th} power, $(-j\omega_\alpha)^N$. Expressing this oriented ramp in terms of the horizontal and vertical ramps provides the basis and coefficients needed to steer derivatives of arbitrary order:

$$(\omega_\alpha)^N = (\cos(\alpha)\omega_x + \sin(\alpha)\omega_y)^N. \qquad (7.35)$$

Although presented in the context of derivatives, the principle of steerability is not limited to derivatives. In the most general case, a two-dimensional filter $f(x,y)$ is steerable in orientation if it can be expressed as a polar-separable function, $g(r)h(\theta)$, where $h(\theta)$ is band-limited. More specifically, for an arbitrary radial component $g(r)$, and for $h(\theta)$ expressed as:

$$h(\theta) \quad = \quad \sum_{n=1}^{N} a_n \cos(n\theta) + b_n \sin(n\theta) \tag{7.36}$$

then the filter is steerable with a basis size of $2N$.

7.4 Edge Detection

Discrete differentiation forms the foundation for many applications in computer vision. One such example is *edge detection* - a topic that has received an excessive amount of attention, but is only briefly touched upon here. An edge is loosely defined as an extended region in the image that undergoes a rapid directional change in intensity. Differential techniques are the obvious choice for measuring such changes. A basic edge detector begins by computing first-order spatial derivatives of an image $f[x,y]$:

$$f_x[x,y] \quad = \quad (f[x,y] \star h'[x]) \star h[y] \tag{7.37}$$
$$f_y[x,y] \quad = \quad (f[x,y] \star h[x]) \star h'[y], \tag{7.38}$$

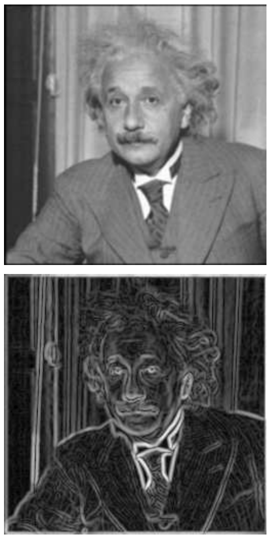
Figure 7.11 Edges

where $h'[\cdot]$ and $h[\cdot]$ are the derivative and prefilter defined in Section 7.2. The "strength" of an edge at each spatial location is defined to be the magnitude of the gradient vector $\bigtriangledown[x,y] = (\, f_x[x,y] \quad f_y[x,y] \,)$, defined as:

$$| \bigtriangledown [x,y]| \quad = \quad \sqrt{f_x^2[x,y] + f_y^2[x,y]}. \tag{7.39}$$

As shown in Figure 7.11, the gradient magnitude is only the beginning of a more involved process (not discussed here) of extracting and localizing the salient and relevant edges.

7.5 Wiener Filter

For any of a number of reasons a digital signal may become corrupted with noise. The introduction of noise into a signal is often modeled as an additive process, $\hat{s} = s + n$. The goal of de-noising is to recover the original signal s from the corrupted signal $\hat{s}$. Given a single constraint in two unknowns this problem is equivalent to my asking you "37 is the sum of two numbers, what are they?" Lacking clairvoyant powers or knowledge of how the individual numbers were selected we have little hope of a solution. But by

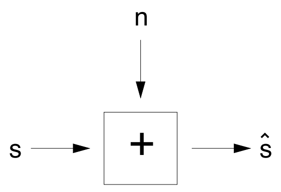

Figure 7.12 Additive noise

making assumptions regarding the signal and noise characteristics and limiting ourselves to a linear approach, a solution can be formulated known as the Wiener filter, of famed Mathematician Norbert Wiener (1894-1964).

Having restricting ourselves to a linear solution, our goal is to design a filter $h[x]$ such that:

$$\begin{aligned} s[x] &= h[x] \star \hat{s}[x] \\ &= h[x] \star (s[x] + n[x]), \end{aligned} \tag{7.40}$$

that is, when the filter is convolved with the corrupted signal the original signal is recovered. With this as our goal, we reformulate this constraint in the frequency domain and construct a quadratic error functional to be minimized:

$$E(H(\omega)) = \int d\omega \, [H(\omega)(S(\omega) + N(\omega)) - S(\omega)]^2. \tag{7.41}$$

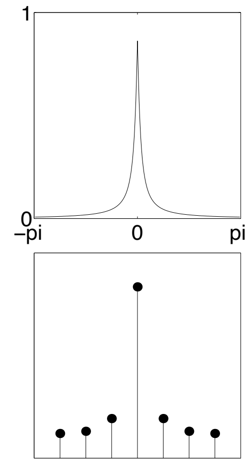

Figure 7.13 Wiener filter

For notational simplicity we drop the frequency parameter ω and express the integral with respect to the expected value operator $\mathcal{E}\{\cdot\}$:

$$\begin{aligned} E(H) &= \mathcal{E}\left\{(H(S+N) - S)^2\right\} \\ &= \mathcal{E}\left\{H^2(S+N)^2 - 2HS(S+N) + S^2\right\} \\ &= \mathcal{E}\left\{H^2(S^2 + 2SN + N^2) - 2H(S^2 + SN) + S^2\right\} \end{aligned} \tag{7.42}$$

In order to simplify this expression we can assume that the signal and noise are statistically independent (i.e., $\mathcal{E}\{SN\} = 0$), yielding:

$$E(H) = \mathcal{E}\left\{H^2(S^2 + N^2) - 2HS^2 + S^2\right\}. \tag{7.43}$$

To minimize, we differentiate:

$$\frac{dE(H)}{dH} = 2H(S^2 + N^2) - 2S^2, \tag{7.44}$$

set equal to zero and solve:

$$H(\omega) = \frac{S^2(\omega)}{S^2(\omega) + N^2(\omega)}. \tag{7.45}$$

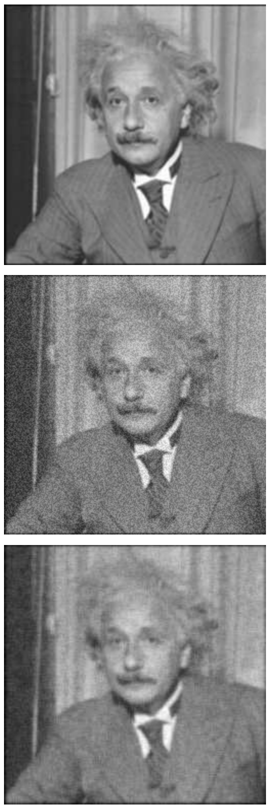

Figure 7.14 Einstein plus noise

At an intuitive level this frequency response makes sense - when the signal is strong and the noise is weak the response is close to 1 (i.e., frequencies are passed), and when the signal is weak and the noise is strong the response is close to 0 (i.e., frequencies are stopped). So we now have an optimal (in the least-squares sense) frequency response in terms of the signal and noise characteristics, but of course we don't typically know what those are. But we can instantiate them by making assumptions about the general statistical nature of the signal and noise, for example a common choice is to assume white noise, $N(\omega)$ is constant for all ω, and, for natural images, to assume that $S(\omega) = 1/\omega^p$. The frequency response in the top panel of Figure 7.13 was constructed under these assumptions. Shown in the bottom panel is a 7-tap filter derived from a least-squares design. This one-dimensional formulation can easily be extended to two or more dimensions. Shown in Figure 7.14 from top to bottom, is Einstein, Einstein plus noise, and the results of applying a 7×7 Wiener filter. Note that the noise levels are reduced but that much of the sharp image structure has also been lost, which is an unfortunate but expected side effect given that the Wiener filter is low-pass in nature.