Chapter 19

Global Operations

If a pixel in the output image \( g \) is a function of (almost) all of the pixels in \( f[x,y] \), then \( O\{f[x,y]\} \) is a global operator. This category includes image coordinate transformations, of which the most important is the Fourier transform. These transformations derive new, usually equivalent, representations of images; for example, the Fourier transform maps from the familiar coordinate-space representation \( f[x,y] \) to a new representation (a new image) whose brightness at each coordinate describes the quantity of a particular sinusoidal spatial frequency component present in \( f[x,y] \). The sum of the component sinusoids is the original image. In other words, the Fourier transform generates the frequency-space representation \( F[\xi,\eta] \) of the image \( f[x,y] \). The coordinates of the image \([x,y]\) have dimensions of length (e.g., mm) while the coordinates of the frequency representation \([\xi,\eta]\) have units of inverse length (e.g., \(\text{cycles/mm}\)). Global gray-level properties of the image map to local properties in the Fourier transform, and vice versa. The frequency-space representation is useful for many applications, including segmentation, coding, noise removal, and feature classification. It also provides an avenue for performing other image operations, particularly convolution. Each output pixel is a function of the gray levels of all input pixels.

19.1 Relationship to Neighborhood Operations

The concept of a linear global operator is a simple extension of that of the linear local neighborhood operator. In that case, an output pixel was calculated by point-by-point multiplication of pixels in the input image by a set of weights (the kernel) and summing the products. The convolution at different pixels is computed by shifting the kernel. Recall that some accommodation must be made for cases where one or more elements of the kernel are off the edge of the image.

In the case of a global operator, the set of weights is as large as the image and constitutes a “mask function”, say \( q[x,y] \). The output value obtained by applying a mask \( q[x,y] \) to an input image \( f[x,y] \) is:

\[
g = \iint f[x,y] \ q[x,y] \ dx \ dy
\]
In the discrete case, the integral becomes a summation:

\[ g = \sum_n \sum_m f[n,m] \cdot q[n,m] \]

Note that a translation of the mask by one pixel in any direction shifts some of its elements over the edge of the image. If we assume that the output in such cases is undefined, only a single output pixel is calculated from one mask function \( q[n,m] \). In general, different outputs result from different masks, i.e., we can define an output pixel by using different masks for each coordinate pair \([x',y']\):

\[ g[k,\ell] = \sum_n \sum_m f[n,m] \cdot q[n,m;k,\ell] \]

Schematic of a global operation evaluated at one pixel \([x',y']\). The input image \( f[x,y] \) is multiplied by the specific “mask function” for that output pixel; the product values are summed to compute the output gray value \( g \).
In general, the coordinates of \( g \) are different from those of \( f \), and often even have different dimensions (units). The action of the operator is obviously determined by the form of the mask function. The most common example is the Fourier transform, where the mask function is:

\[
q[x, y; \xi, \eta] = \cos[2\pi(\xi x + \eta y)] - i \sin[2\pi(\xi x + \eta y)] = \exp[-2\pi i(\xi x + \eta y)]
\]

### 19.2 Discrete Fourier Transform (DFT)

If the input signal has been sampled at discrete intervals (of width \( \Delta x \), for example), the Fourier integral over \( x \) reduces to a sum:

\[
F[\xi] = \sum_{n=-\infty}^{+\infty} f[n \cdot \Delta x] \exp[-2\pi i n \cdot \Delta x] \quad \xi = \frac{\xi_N}{X_{\text{min}}} = \frac{1}{2 \cdot \Delta x}
\]

Recall that the Whittaker-Shannon sampling theorem states that a sinusoidal function must be sampled at a rate greater than than two samples per period (Nyquist frequency) to avoid aliasing. Thus, the minimum period \( X_{\text{min}} \) of a sampled sinusoidal function is two sample intervals (\( 2 \cdot \Delta x \) in the example above), which implies that the maximum spatial frequency in the sampled signal is:

\[
\xi_{\text{max}} = \xi_{N_{\text{yq}}} = \frac{1}{X_{\text{min}}} = \frac{1}{2 \cdot \Delta x}
\]
\( \xi_{\text{max}} \) is measured in cycles per unit length (typically cycles per millimeter). Often the absolute scale of the digital image is not important, and the frequency is scaled to \( \Delta x = 1 \) pixel, i.e., the maximum spatial frequency is \( \frac{1}{2} \) cycle/pixel. The range of meaningful spatial frequencies of the DFT is \( \frac{1}{2 \Delta x} > |\xi| \).

If the input function \( f[x] \) is limited to \( N \) samples, the DFT becomes a finite sum:

\[
F[\xi] \equiv \sum_{n=0}^{N-1} f[n \cdot \Delta x] \exp[-2\pi i \xi (n \cdot \Delta x)]
\]

or

\[
\sum_{n=-N/2}^{N/2-1} f[n \cdot \Delta x] \exp[-2\pi i \xi (n \cdot \Delta x)]
\]

The DFT of a 1-D sequence of \( N \) samples at regular intervals \( \Delta x \) can be computed at any spatial frequency \( \xi \). However, it is usual to calculate the DFT of a sequence of frequencies (e.g., a total \( M \)) separated by a constant interval \( \Delta \xi \). Each sample of the DFT of a real sequence of \( N \) pixels requires that \( N \) values each of the cosine and sine be computed, followed by \( 2N \) multiplications and \( 2N \) sums, i.e., of the order of \( N \) operations. The DFT at \( M \) spatial frequencies requires of the order of \( M \cdot N \) operations. Often, the DFT is computed at \( N \) frequencies, thus requiring of the order of \( N^2 \) operations. This intensity of computation made calculation of the DFT a tedious and rarely performed task before digital computers. For example, a Fourier deconvolution of seismic traces for petroleum exploration was performed by Enders Robinson in 1951; it took the whole summer to do 32 traces by hand with a memoryless mechanical calculator. This task could now be done with the cheapest PC in less than a second. Even with mainframe digital computers into the 1960s, digital Fourier analysis was unusual because of the computation time. In 1965, J.W. Cooley and J.W. Tukey developed the Fast Fourier Transform algorithm, which substantially cut computation times and made digital Fourier analysis feasible.

### 19.3 Fast Fourier Transform (FFT)

The FFT was developed to compute discrete Fourier spectra with fewer operations than the DFT by sacrificing some flexibility. The DFT may compute the amplitude of sinusoidal components at any frequency within the Nyquist window, i.e., the DFT maps discrete coordinates \( n \cdot \Delta x \) to a continuous set of frequencies \( \xi \) in the interval \( [-\xi_{\text{Nyq}}, \xi_{\text{Nyq}}] \). The DFT may be computed at a single spatial frequency if desired. The FFT is a recursive algorithm that calculates the spectrum at a fixed discrete set of frequencies with a minimum number of repetitive calculations. The spectrum must be computed at all frequencies to obtain the values of individual spectral components. In the FFT, the amplitudes at \( N \) discrete equally spaced frequencies are computed in the interval \( [-\xi_{N\text{Nyq}}, \xi_{N\text{Nyq}}] \) from \( N \) input samples. The frequency samples are indexed
by the integer $k$ and the interval between frequency samples is:

$$\Delta \xi = \frac{1}{N} \cdot 2 \xi_{Nyq} = \frac{1}{N} \cdot \frac{2}{2 \cdot \Delta x} = \frac{1}{N \cdot \Delta x}$$

$$\implies \xi_k = \frac{k}{N \cdot \Delta x}, \left[ -\frac{N}{2} \leq k \leq \frac{N}{2} - 1 \right]$$

$$\implies N \cdot \Delta x \cdot \Delta \xi = 1$$

If we substitute these specific frequencies into the DFT:

$$F \left[ k \cdot \Delta \xi \right] = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} f \left[ n \cdot \Delta x \right] \exp \left[ -2\pi i k \cdot \Delta \xi \cdot (n \cdot \Delta x) \right]$$

$$= \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} f \left[ n \cdot \Delta x \right] \exp \left[ -2\pi i k n \cdot \frac{\Delta x}{N \cdot \Delta x} \right]$$

but $\Delta \xi = \frac{1}{N \cdot \Delta x} \implies F \left[ k \cdot \Delta \xi \right] = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} f \left[ n \cdot \Delta x \right] \exp \left[ -\frac{2\pi i n}{N} \right]$}

If $\Delta x$ is assumed to be a dimensionless sample, then the Nyquist frequency is fixed at:

$$\xi_{Nyquist} = \frac{1}{2} \text{ cycle/sample} = \frac{\pi}{2} \text{ radians/sample}$$

Recall that the DFT assumes that the sample interval is $\Delta x$ and computes a periodic spectrum with period $\frac{1}{\Delta x}$. In the FFT, the spectrum is assumed to be sampled at intervals $\Delta \xi = \frac{1}{N \cdot \Delta x}$, which implies in turn that the input function is periodic with period $N \cdot \Delta x$. If $N$ is a power of 2 (e.g., 128, 256, 512, $\cdots$), there are only $N$ distinct values of the complex exponential $\exp \left[ -\frac{2\pi i n}{N} \right]$ to calculate. By using this fact, the number of required operations may be reduced and processing speeded up. The FFT of $N$ samples requires of the order $N \cdot \log_2 [N]$ operations vs. $O \{N^2\}$ for the DFT.

Since both representations $f[n \cdot \Delta x]$ and $F[k \cdot \Delta \xi]$ are sampled and periodic, the inverse FFT is a finite summation and is proportional to:

$$f \left[ n \cdot \Delta x \right] = C \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} F \left[ k \cdot \Delta \xi \right] \exp \left[ +\frac{2\pi i n k}{N} \right]$$

The proportionality constant $c$ is required to ensure that $\mathcal{F}_1 \{F[k]\} = f[n]$, and may be found by substituting the formula for the forward FFT for $F[k \cdot \Delta \xi]$:
\[ f[n \cdot \Delta x] = C |\Delta x| \sum_{k=-N/2}^{N/2-1} \left( \sum_{m=-N/2}^{N/2-1} f[m \cdot \Delta x] \exp \left[ -\frac{2\pi i nk}{N} \right] \right) \exp \left[ +\frac{2\pi i nk}{N} \right] \]

\[
= C \sum_{k=-N/2}^{N/2-1} \sum_{m=-N/2}^{N/2-1} f[m \cdot \Delta x] \exp \left[ -\frac{2\pi i k}{N} (n - m) \right] \]

\[
= C \sum_{m=-N/2}^{N/2-1} f[m \cdot \Delta x] \sum_{k=-N/2}^{N/2-1} \exp \left[ -\frac{2\pi i k}{N} (n - m) \right] \]

\[
= C \sum_{m=-N/2}^{N/2-1} f[m \cdot \Delta x] \cdot (N \cdot \delta[n - m]) \]

\[
= C \cdot N \cdot f[n \cdot \Delta x] \]

\[
= f[n \cdot \Delta x] \]

Thus \( C = N^{-1} \) and the inverse FFT may be defined as:

\[ f[n \cdot \Delta x] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} F[k \cdot \Delta \xi] \exp \left[ +\frac{2\pi i nk}{N} \right] \]

The proportionality constant is a scale factor that is only significant when cascading forward and inverse transforms and may be applied in either direction. Many conventions (including mine) include the proportionality constant in the forward FFT:

\[ F[k \cdot \Delta \xi] = F \left[ \frac{k}{N \cdot \Delta x} \right] = \frac{1}{N} \sum_{n=-N/2}^{N/2-1} f[n \cdot \Delta x] \exp \left[ -\frac{2\pi i nk}{N} \right] \]

\[ f[n \cdot \Delta x] = \sum_{k=-N/2}^{N/2-1} F[k \cdot \Delta \xi] \exp \left[ +\frac{2\pi i nk}{N} \right] \]

\[ N \cdot \Delta x \cdot \Delta \xi = 1 \]

\[ \xi_{Nyq} = N \cdot \frac{\Delta \xi}{2} \]

\[ x_{\text{max}} = N \cdot \frac{\Delta x}{2} \]
19.4 DFTs of Images

The concept of a 1-D Fourier transform can be easily extended to multidimensional continuous or discrete signals. The continuous 2-D transform is defined as:

\[
F_2 \{ f[x,y] \} \equiv F[\xi,\eta] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f[x,y] \exp[-2\pi i (\xi x + \eta y)] \, dx \, dy
\]

For a uniformly sampled discrete 2-D function \( f[n,m] \), the transform is a summation:

\[
F_2 \{ f[n \cdot \Delta x, m \cdot \Delta y] \} = \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} f[n,m] \exp[-2\pi i (\xi n \cdot \Delta x + \eta m \cdot \Delta y)] = F[\xi,\eta]
\]

The Fourier transform of a real-valued 2-D function is Hermitian (even real part and odd imaginary part). The Fourier transform of the image of a 2-D cosine is a pair of delta-function spikes at a distance from the origin proportional to the frequency of the cosine, as shown on the next page. The polar angle of the spikes relative to the origin indicates the direction of variation of the cosine, while the brightness of the spikes is proportional to the amplitude of the cosine. Notice that the Fourier transform of the sum of two cosine waves is the sum of the individual transforms (i.e., the Fourier transform is linear).

If the sampled input image \( f[n,m] \) has \( N \times N \) pixels, then there are only \( N^2 \) pieces of information in the input. Thus the DFT \( F[\xi,\eta] \) also must contain at most only \( N^2 \) independent pieces of information. This allows the DFT to be sampled without loss of information:

\[
F[\xi,\eta] \rightarrow F[k \cdot \Delta \xi, \ell \cdot \Delta \eta] \rightarrow F[k,\ell]
\]

It is easy to show (see linear math course) that

\[
\Delta \xi = \frac{1}{N \cdot \Delta x}
\]
\[
\Delta \eta = \frac{1}{N \cdot \Delta y}
\]

The 2-D transform has the same properties mentioned before, including that global properties become local properties and vice versa. This is the primary reason why the Fourier transform is such a powerful tool for image processing and pattern recognition; \( F[\xi,\eta] \) is uniquely defined for each \( f[x,y] \), and the global properties of \( f[x,y] \) are concentrated as local properties of \( F[\xi,\eta] \). Therefore:

1. local modification of \( f[n,m] \) \( \implies \) global modification of \( F[k,\ell] \)
2. global modification of \( f[n,m] \) \( \implies \) local modification of \( F[k,\ell] \)

Local modification in the space domain is what we call “filtering” and is intimately
related the local operation of convolution that we’ve already discussed. In fact, it is easy to prove that the Fourier transform of a convolution is the product of the Fourier transforms of the component functions. This result is called the filter theorem.

\[ \mathcal{F}_2 \{ f[n,m] * h[n,m] \} = \mathcal{F}_2 \{ f[n,m] \} \cdot \mathcal{F}_2 \{ h[n,m] \} \]

\[ \Longrightarrow f * h = \mathcal{F}_2^{-1} \{ F[k,\ell] \cdot H[k,\ell] \} \]

We have already given the name impulse response or point spread function to \( h[x,y] \); the representation \( H[\xi,\eta] \) is called the transfer function of the system. The most common reason for computing Fourier transforms of digital signals or images is the to use this path for convolution. Examples are shown for both lowpass filtering and differentiation (highpass filtering).

Averaging of a bitonal “E” in the vertical direction with \( h[n,m] = \delta_d[n] \cdot RECT \left( \frac{m}{m} \right) \); the transfer function is “skinny” in the \( \eta \)-direction, thus attenuating sinusoidal components that oscillate in the vertical direction.
Differentiation in the x-direction of a pair of “E”s applied in the Fourier domain. The impulse response is \( h[n, m] = \delta_d[n + 1, m] - \delta_d[n, m] \); the MTF is large for \(|k| >> 0\), so it enhances high-frequency sinusoids that oscillate in the horizontal direction. The output shows just the vertical edges of the “E”s.
19.5 Other Global Operations

Other families of mask functions may be used in the general coordinate transformation equation:

\[
g[u] = \int_{-\infty}^{+\infty} f[x] \ q[x;u] \ dx \quad (1-D \ continuous \ functions)
\]

\[
g[k] = \sum_n f[n] \ q[n;k] \quad (1-D \ discrete \ functions)
\]

\[
g[u,v] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f[x,y] \ q[x,y;u,v] \ dx \ dy \quad (2-D \ continuous \ functions)
\]

\[
g[k,\ell] = \sum_n \sum_m f[n,m] \ q[n,m;k,\ell] \quad (2-D \ discrete \ functions)
\]

Such a transformation is invertible if \( f[x,y] \) can be derived from \( g[u,v] \) via an expression of the form:

\[
f[x] = \int_{-\infty}^{+\infty} g[u] \ q'[x;u] \ du \quad (1-D \ continuous \ functions)
\]

\[
f[n] = \sum_n g[k] \ q'[n;k] \quad (1-D \ discrete \ functions)
\]

\[
f[x,y] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g[u,v] \ q'[x,y;u,v] \ du \ dv \quad (2-D \ continuous \ functions)
\]

\[
f[n,m] = \sum_k \sum_\ell g[k,\ell] \ q'[n,m;k,\ell] \quad (2-D \ discrete \ functions),
\]

where \( m' \) is the 4-D mask function for the inverse transform. For the purposes of this course it is not really essential to understand the conditions for the transformation to be invertible, so we will just say that the set of mask functions \( q[x,y;u,v] \) must be complete, i.e., any function \( f[x,y] \) must be representable as a sum (linear combination) of the mask functions.

19.6 Discrete Cosine Transform (DCT)

The discrete cosine transform (DCT) is a relative of the discrete Fourier transform that is often used in digital image coding or compression. The term is used to denote any of several related algorithms, but we will consider only the even symmetric DCT. Consider a 1-D sampled image \( f[n] \) with \( N \) pixels in the interval \( 0 \leq n \leq N - 1 \). Recall that the DFT assumed that the \( N \)-pixel image \( f[n] \) is actually of infinite extent but is periodic over \( N \) samples, as below:
Arrangement of pixel blocks in discrete Fourier transform (DFT) and discrete cosine transform (DCT). Pixel blocks in the DFT are strictly periodic, thus possibly producing large discontinuities in gray value at the edges; pixel blocks in the DCT are reversed, replicating gray values across the edges of blocks and reducing the discontinuities at the edges.

The DCT of $f[n]$ builds a new function $g[n]$ that is periodic over $2N$ samples in the interval $0 \leq 2N - 1$:

$$g[n] = f[n], \quad 0 \leq n \leq N - 1$$

$$g[n] = f[2N - 1 - n], \quad N \leq n \leq 2N - 1$$

The DCT is the $2N$-point DFT of $g[n]$. In this manner, the discontinuities at the edges of the image are eliminated, and thus removing leakage from the transform representation. The energy in the transform will not spread over so many pixels, and thus the DCT representation will be more compact than the DFT. In addition, the $2N$-point DCT will be real because the new function $g[n]$ is intrinsically even.

The 2-D DCT is a simple extension of the 1-D version. Given an $N \times N$ image $f[n,m]$, a $2N \times 2N$ version $g[n,m]$ is created by replication and reflection. The new image will be smoothly periodic over $2N \times 2N$ so that leakage will not be present.

To reduce the impact of the sharp transitions that often occur at the edges of blocks, as well as to obtain a transform that is real-valued for a real-valued input image, the discrete cosine transform (DCT) is used instead of the DFT. The DCT has become very important in the image compression community, being the basis transformation for the JPEG and MPEG compression standards. The DCT of an $M \times M$ block may be viewed as the DFT of a synthetic $2M \times 2M$ block that is created by replicating the original $M \times M$ block after folding about the vertical and horizontal edges:
The original $4 \times 4$ block of image data is replicated $4$ times to generate an $8 \times 8$ block of data via the DFT format and an $8 \times 8$ DCT block by appropriate reversals. The transitions at the edges of the $4 \times 4$ DCT blocks do not exhibit the “sharp” edges in the $4 \times 4$ DFT blocks.

The edge discontinuities of the resulting $2M \times 2M$ block have smaller amplitudes. The symmetries of the Fourier transform for a real-valued image ensure that the original $M \times M$ block may be reconstructed from the DCT of the $2M \times 2M$ block.

Consider the computation of the DCT for a 1-D $M$-pixel block $f[n]$ $(0 \leq n \leq M - 1)$. The $2M$-pixel synthetic array $g[n]$ is indexed over $n$ $(0 \leq n \leq 2M - 1)$ and has the form:

$$g[n] = \begin{cases} f[n] & \text{if } 0 \leq n \leq M - 1 \\ f[2M - 1 - n] & \text{if } M \leq n \leq 2M - 1 \end{cases}$$

In the case $M = 8$, the array $g[n]$ is defined:

$$g[n] = \begin{cases} f[n] & \text{if } 0 \leq n \leq 7 \\ f[15 - n] & \text{if } 8 \leq n \leq 15 \end{cases}$$
The values of $g[n]$ for $8 \leq n \leq 15$ is a “reflected replica” of $f[n]$:

- $g[8] = f[7]$
- $g[9] = f[6]$
- $g[10] = f[5]$
- $g[12] = f[3]$
- $g[14] = f[1]$
- $g[15] = f[0]$

If the “new” array $g[n]$ is assumed to be periodic over $2M$ samples, its amplitude is defined for all $n$, e.g.,

$$g[n] = \begin{cases} 
  f[-1-n] & \text{if } -M \leq n \leq -1 \implies -16 \leq n \leq -1 \\
  f[n+2M] & \text{if } -2M \leq n \leq -M-1 \implies -32 \leq n \leq -17
\end{cases}$$

Note that the 16-sample block $g[n]$ is NOT symmetric about the origin of coordinates because $g[-1] = g[0]$; to be symmetric, $g[-\ell]$ would have to equal $g[+\ell]$. For example, consider a 1-D example where $f[n]$ is an 8-pixel ramp as shown:
DFT of an 8-pixel “ramp” (a) \( f_1[n] = n \cdot \text{STEP}[n] \) for \( 2N = 16 \); (b) \( \Re \{ F_1[k] \} \); (c) \( \Im \{ F_1[k] \} \), showing the redundancy of the complex spectrum and the high-frequency terms due to the discontinuous transition at the edge.

The 2\( M \)-point representation of \( f[n] \) is the \( g[n] \) just defined:

\[
g[n] = \begin{cases} 
  f[n] & \text{for } 0 \leq n \leq 7 \\
  f[2M - 1 - n] & \text{for } 8 \leq n \leq 15 
\end{cases}
\]

If this function were symmetric (even), then circular translation of the 16-point array by 8 pixels to generate \( g[n - 8 \mod 16] \) also be an even function.

From the graph, it is apparent that the translated array is not symmetric about the origin; rather, it has been translated by \(-\frac{1}{2}\) pixel from symmetry in the 2\( M \)-pixel array. Thus define a new 1-D array \( c[n] \) that is shifted to the left by \( \frac{1}{2} \) pixel:

\[
c[n] = g \left[ n - \frac{1}{2} \right]
\]
This result may seem confusing at first; how can a sampled array be translated by \( \frac{1}{2} \) pixel? For the answer, consider the continuous Fourier transform of a sampled array translated by \( \frac{1}{2} \) unit:

\[
\mathcal{F}_1 \{ c[n] \} \equiv C[\xi] = \mathcal{F}_1 \left\{ g \left[ x - \frac{1}{2} \right] \right\} = \mathcal{F}_1 \left\{ g[x] * \delta \left[ x - \frac{1}{2} \right] \right\} = G[\xi] \cdot \exp \left[ -2\pi i \xi \cdot \frac{1}{2} \right] C[\xi] = G[\xi] \cdot \exp \left[ -i\pi \xi \right]
\]

Thus the effect of translation by \( \frac{1}{2} \) pixel in the space domain is multiplication of the Fourier transform by the specific linear-phase factor:

\[
\exp \left[ -i\pi \xi \right] = \cos \left[ \pi \xi \right] - i \sin \left[ \pi \xi \right].
\]

The \( 2M \)-point DFT of the symmetric discrete array (original array translated by \( \frac{1}{2} \) pixel) has the form:

\[
F_{2M} \{ c[n] \} = \mathcal{F}_{2M} \left\{ g \left[ n - \frac{1}{2} \right] \right\} = \mathcal{F}_{2M} \left\{ g[n] * \delta \left[ n - \frac{1}{2} \right] \right\} = G[k] \cdot \exp \left[ -i\pi \left( \frac{k}{2M} \right) \right] C[k] = G[k] \cdot \exp \left[ -i\pi \frac{k}{2M} \right] = G[k] \cdot \left( \cos \left[ \frac{\pi k}{2M} \right] - i \sin \left[ \frac{\pi k}{2M} \right] \right)
\]

where the continuous spatial frequency \( \xi \) has been replaced by the sampled frequency \( \frac{k}{2M} \). This function \( C[k] \) is the DCT of \( f[n] \). Because the \( 2M \)-point translated function \( c[n] \) is real and even, so must be the \( 2M \)-point discrete spectrum \( C[k] \); therefore only \( M \) samples of the spectrum are independent. This array is the DCT of \( f[n] \).
Discrete cosine transform of ramp function is equivalent to the DFT of the symmetric function shown in (a) $f_2[n] = f_1[n] + f_1[-(n + 1)]$ for $2N = 16$; (b) $\Re \{ F_2[k] \}$; (c) $\Im \{ F_2[k] \}$, showing the reduction in the relative amplitude of the imaginary part compared to $F_1[k]$.

19.6.1 Steps in Forward DCT

To summarize, the steps in the computation of the 1-D DCT of an $M$-point block $f[n]$ are:

1. create a $2M$-point array $g[n]$ from the $M$-point array $f[n]$:

   $g[n] = f[n]: 0 \leq n \leq M - 1$
   $g[n] = f[2M - 1 - n]: M \leq n \leq 2M - 1$

2. compute the $2M$-point DFT of $g[n] = G[k]$

3. the $M$-point DCT $C[k] = \exp \left[ -i\frac{\pi k}{2M} \right] \cdot G[k]$ for $0 \leq k \leq M - 1$
The entire process may be cast into the form of a single equation, though the algebra required to get there is a bit tedious,

\[
C[k] = \sum_{n=0}^{M-1} 2 f[n] \cos \left( \frac{\pi k \cdot 2n + 1}{2M} \right) \quad \text{for} \quad 0 \leq k \leq M - 1
\]

19.6.2 Steps in Inverse DCT

The inverse DCT is generated by applying the procedures in the opposite order:

1. create a 2M-point array \( G[k] \) from the M-point DCT \( C[k] \):

\[
G[k] = \exp \left[ + \frac{i \pi k}{2M} \right] \cdot C[k] \quad \text{for} \quad 0 \leq k \leq M - 1
\]

\[
G[k] = - \exp \left[ + \frac{i \pi k}{2M} \right] \cdot C[2M - k] \quad \text{for} \quad M + 1 \leq k \leq 2M - 1
\]

2. compute the inverse 2M-point DFT of \( G[k] \rightarrow g[n] \)

3. \( f[n] = g[n] \) for \( 0 \leq n \leq M - 1 \)

The single expression for the inverse DCT is:

\[
f[n] = \frac{1}{M} \sum_{k=0}^{M-1} w[k] \cdot C[k] \cos \left( \frac{\pi k \cdot 2n + 1}{2M} \right) \quad \text{for} \quad 0 \leq n \leq M - 1
\]

where \( w[k] = \frac{1}{2} \) for \( k = 0 \) and \( w[k] = 1 \) for \( 1 \leq k \leq M - 1 \).

19.7 Walsh-Hadamard Transform

A transform which has proven useful in image compression and pattern recognition grew out of matrix theory and was first described by Hadamard in 1893. A modification made by Walsh in 1923 is commonly used, and so the transformation is often named after both men. The W-H transform resembles a Fourier transform with a mask that has been thresholded so that there are only two values, \( \pm 1 \). We will first consider the W-H transform of a one-dimensional function \( f[n] \). The 1-D W-H transform of a two-pixel image is derived from two mask functions which may be represented as vectors:

\[ g[k] = \sum_{n=0}^{1} f[n] \ q[n; k] \quad \text{1-D discrete functions} \]

\[ q[0, 0] = +1, \ q[0, 1] = +1 \implies \mathbf{q}_0 = \begin{bmatrix} +1 \\ +1 \end{bmatrix} \]

\[ q[1, 0] = +1, \ q[1, 1] = -1 \implies \mathbf{q}_1 = \begin{bmatrix} +1 \\ -1 \end{bmatrix} \]

Thus the elements of the 2-pixel W-H transform are the sum and difference of the pixel gray values, which is identical to the 2-pixel Fourier transform. The vectors \( m_0 \) and \( m_1 \) are orthogonal (i.e., perpendicular) so that:

\[ \mathbf{q}_i \cdot \mathbf{q}_j \equiv \left( \mathbf{q}_i \right)_0 \cdot \left( \mathbf{q}_j \right)_0 + \left( \mathbf{q}_i \right)_1 \cdot \left( \mathbf{q}_j \right)_1 = \begin{cases} 0 & \text{if } i \neq j \\ 2 & \text{if } i = j \end{cases} \]

The column vectors that define the mask can be assembled into a 2 × 2 orthogonal matrix:

\[
\mathbf{H}_2 \equiv \begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \end{bmatrix} = \begin{bmatrix} +1 \\ +1 \\ +1 \\ -1 \end{bmatrix} = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}
\]

\[
\mathbf{H}_2^T = \mathbf{H}_2
\]

\[
\mathbf{H}_2 \cdot \mathbf{H}_2^T = (H_2)^2 = \begin{bmatrix} +2 & 0 \\ 0 & +2 \end{bmatrix} = 2\mathbf{I}
\]

where \( \mathbf{I} \) is the 2 × 2 identity matrix. Note that \( \mathbf{H}_2 \) is identical to its transpose \( \mathbf{H}_2^T \), which is the criterion that defines an orthogonal matrix.

The inverse matrix \( \mathbf{H}_2^{-1} \) is defined as the matrix that satisfies:

\[
\mathbf{H}_2 \mathbf{H}_2^{-1} = \mathbf{I}
\]

which shows that the inverse W-H transform is proportional to the transpose of \( \mathbf{H}_2 \).
and thus to $H_2$ itself.

\[
H_2 = \begin{bmatrix}
+1 & +1 \\
+1 & -1
\end{bmatrix}
\]

\[
H_2^{-1} = \frac{1}{2} \begin{bmatrix}
+1 & +1 \\
+1 & -1
\end{bmatrix}
\]

The four-pixel W-H matrix is obtained by the direct product of the two-pixel W-H matrix with itself, i.e.,

\[
H_4 = H_2 \times H_2 = \begin{bmatrix}
H_2 & H_2 \\
H_2 & -H_2
\end{bmatrix} = \begin{bmatrix}
+1 & +1 & +1 & +1 \\
+1 & -1 & -1 & -1 \\
+1 & +1 & -1 & -1 \\
+1 & -1 & -1 & +1
\end{bmatrix}
\]

Thus when applied to a 4-pixel image with values $f(n)$ for $0 \leq n \leq 3$, the W-H transform is a 4-pixel image with gray values:

\[
\]

\[
\]

\[
\]

\[
\]

Note that the elements of $w[k]$ are sums and differences of the elements of $f[n]$, and thus no multiplication is required. Also the elements of the W-H transform representation will be integers if so are the input gray values. This is the source of one useful characteristic of the W-H transform: that the transform representation need not be requantized for display. Also note that the elements of the mask functions of the W-H transform are all purely real numbers so that the W-H transform of a real image is real.

The inverse 4-pixel W-H transform is easily confirmed to be:

\[
f[0] = \frac{1}{4}(w[0] + w[1] + w[2] + w[3])
\]

\[
f[1] = \frac{1}{4}(w[0] - w[1] + w[2] - w[3])
\]

\[
\]

\[
\]

Recall that the elements of the Fourier transform are ordered in terms of the spatial
frequency of the mask functions. In similar fashion, the W-H transform elements can be ordered in terms of the number of sign changes of the mask function, which is called the sequency of the term. The *sequency-ordered* Walsh-Hadamard transformation matrix is:

\[
H'_4 = \begin{bmatrix}
+1 & +1 & +1 & +1 \\
+1 & +1 & -1 & -1 \\
+1 & -1 & -1 & +1 \\
+1 & -1 & +1 & -1 \\
\end{bmatrix}
\]

(no sign changes)  
(one sign change)  
two sign changes)  
(three sign changes)

Note that the ordered W-H matrix is also orthogonal, so that the ordered inverse W-H transform is proportional to the ordered forward transform. In similar fashion, the unordered matrix \(H_8\) can be obtained by computing the direct product of \(H_4\) and \(H_2\):

\[
H_8 = H_4 \times H_2 = \begin{bmatrix}
H_2 & +H_2 & +H_2 & +H_2 \\
+H_2 & +H_2 & -H_2 & -H_2 \\
+H_2 & -H_2 & -H_2 & +H_2 \\
+H_2 & -H_2 & +H_2 & -H_2 \\
\end{bmatrix}
\]

\[
= H_2 \times H_4 = \begin{bmatrix}
+1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\
+1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\
+1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\
+1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 \\
\end{bmatrix}
\]

The sequency-ordered matrix is:
\begin{align*}
\mathbf{H}_s' = (\mathbf{H}_s')^T = \\
\begin{bmatrix}
\begin{array}{cccccccc}
+1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\
+1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\
+1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\
+1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\
+1 & -1 & -1 & +1 & +1 & -1 & +1 & +1 \\
+1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 \\
+1 & -1 & +1 & -1 & +1 & -1 & +1 & -1
\end{array}
\end{bmatrix}
\end{align*}

Basis functions of the Walsh-Hadamard transform arranged in order of increasing
“sequency,” which is analogous to the “frequency” of the sinusoidal basis functions of the Fourier transform. An 8-sample 1-D input function is decomposed into how much of that function can be written as each of these basis functions.

19.7.1 Interpretation of the W-H Transform

Note that the W-H matrices may be normalized so that the forward and inverse transforms are completely identical. This is done by dividing the elements of the matrix by $\sqrt{N}$ so that $H_N \cdot H_N = I$.

The relative sizes of elements of the W-H representation will indicate the busyness of the input image; a smooth image will have larger values of the W-H transform for small values of $k$ while the W-H transform of a busy image will be larger for larger values of $k$. The gray value of an element of the transform may be interpreted as the similarity between the input image and the mask image. The elements of the W-H transform of realistic images whose gray levels are well-correlated (i.e., smooth) will tend to be large for small values of $k$. In other words, the energy of the transformed image will tend to be compressed into the pixels indexed by small $k$; pixels with large $k$ will have small values. This property is called energy compaction, and is useful in (and in fact, is the whole basis for) signal compression.
Examples of the 1-D Walsh-Hadamard transform evaluated over 64 pixels. As the input image gets “busier,” the maximum sequency of the W-H transform increases.

The set of mask functions for the 2-D W-H transform are products of the 1-D functions, i.e., the transform is separable. The $8 \times 8$ basis functions and the decomposition of two gray-scale images into their $8 \times 8$ block Walsh-Hadamard transforms are shown.
The 64 basis functions of the $8 \times 8$ Walsh-Hadamard transform: the 64 basis functions are shown on the left and are segmented by red dashed lines. The resulting $8 \times 8$ W-H transforms are on the right, and show that only one pixel is positive in each.

Two examples of $8 \times 8$ block Walsh-Hadamard transforms, showing that busier parts of the image produce more outputs with larger sequencies.