1 Introduction

This chapter is concerned the definition, interpretation and application of Discrete Fourier Transform (DFT) techniques in multi-media filtering and content analysis. If you are not already intimately familiar with the one dimensional DFT and its fast implementation, the FFT, you should carefully review Section 2. Amongst other things, this section supplies the vital connection between the DFT and other Fourier transforms, along with the method for implementing true convolution in the DFT domain. These ideas are extended to multiple dimensions in Section 3, most of which is straightforward. Section 4 shows how the FFT plays an important role in efficient pattern matching, considering both correlation (matched filtering) and normalized cross correlation matching functions and their interpretation. These methods are frequently used to search for features of interest within images and 3D volumes.

Finally, Section 5 considers the problem of power spectrum estimation. In a later chapter, we shall see that power spectrum estimation is a key tool in the analysis of texture content in images. Power spectrum estimation also underlies many filter design strategies, since filter design objectives must generally be met best in those regions of the spectrum where the signal is expected to have significant power. The discussion in Section 5 is rather lengthy, mainly because some effort is devoted to first identifying what is meant by a power spectrum. In a first reading of these notes, it would be possible to largely skip over the material in Section 5.1.

2 The DFT in One Dimension

2.1 Definition

Recall that the Discrete Time Fourier Transform (DTFT) is defined on sequences of infinite extent according to

\[ \hat{x}(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-jn\omega}, \quad \text{for } -\pi \leq \omega \leq \pi \]  

(1)
with the inverse DTFT given by

\[ x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} d\omega \cdot \hat{x}(\omega)e^{jn\omega} \]  

(2)

Although \( x[n] \) is a sequence defined only at integer coordinates \( n \), its DTFT \( \hat{x}(\omega) \) is a continuous function over \( \omega \in [-\pi, \pi] \). Moreover, if \( x[n] \) is obtained by Nyquist sampling a continuous signal \( x(t) \), \( \hat{x}(\omega) \) is both the DTFT of \( x[n] \) and the FT (Fourier Transform) of \( x(t) \).

By contrast, the Discrete Fourier Transform (DFT) involves a finite number of discrete frequencies. It is defined only for sequences \( x[n] \) of finite support, e.g. \( 0 \leq n < N \), where \( N \) is the length of the sequence. It is not necessary for us to start the indices \( n \) from 0, but this is the most common convention. With this convention, the \( N \)-point DFT and its inverse are given by the following equations:

\[
X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi nk}{N}}, \quad 0 \leq k < N
\]

\[
x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi nk}{N}}, \quad 0 \leq n < N
\]

The complex exponentials involved in the forward and inverse DFT are used so commonly in signal processing that it becomes convenient to define the special symbol,

\[ W_N^k \triangleq e^{j\frac{2\pi k}{N}} \]

Noting that \( W_N^k = \left( e^{j\frac{2\pi}{N}}\right)^k = e^{j\frac{2\pi nk}{N}} \), the DFT and its inverse may be expressed as

\[
X[k] = \sum_{n=0}^{N-1} x[n]W_N^{-nk}
\]

\[
x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]W_N^k
\]

Note also that \( \{W_N^k\}_{k=0,1,...,N-1} \) are the \( N \) complex roots of unity.

We may also express the DFT in vector notation as follows. Let \( w_k \) denote the vector (sequence) whose elements are

\[ w_k[n] = W_N^k = e^{j\frac{2\pi nk}{N}} \]

Then the DFT may be expressed as

\[ X[k] = \langle x, w_k \rangle \]
and the inverse DFT may be expressed as

$$x = \frac{1}{N} \sum_{k=0}^{N-1} X[k] w_k$$

This formulation provides great insight into the DFT operation. It indicates that the vectors $w_k$ form a basis for the set of all length $N$ input sequences, $x$. Moreover, this basis is orthogonal, so that $\langle w_i, w_j \rangle = 0$ for $i \neq j$.

It can sometimes be more insightful to work with orthonormal basis sets. To this end, define

$$w'_k = \frac{1}{\sqrt{N}} w_k$$

These form an orthonormal basis for the set of input sequences, as may be seen by rewriting the DFT in a normalized form, with

$$X'[k] = \frac{1}{\sqrt{N}} X[k] = \langle x, w'_k \rangle$$

and

$$x = \sum_{k=0}^{N-1} X'[k] w'_k$$

$$= \sum_{k=0}^{N-1} \langle x, w_k \rangle \cdot w_k$$

In this form we can clearly see that the input vector (sequence) $x$ is just being written as the sum of its projections, $\langle x, w'_k \rangle \cdot w'_k$, onto the orthonormal basis vectors, $w'_k$.

### 2.2 Relationship Between DFT and DTFT

The DFT and DTFT are closely related; in fact, under appropriate conditions, they are equivalent, which is the foundation for much of the usefulness of the DFT. Let $x[n]$ denote a finite support sequence, which is equal to zero everywhere except in the region, $0 \leq n < N$. The $N$-point DFT of the finite sequence, restricted to this region may be written

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi nk}{N}}$$

$$= \left( \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \right) \bigg|_{\omega = \frac{2\pi k}{N}}$$

So the DFT $X[k]$ is a sampling of the spectrally continuous DTFT $\hat{x}(\omega)$ at the locations $\omega = \frac{2\pi k}{N}$. 
Although this relationship is mathematically correct, it violates our convention that $\hat{x}(\omega)$ is defined only on $-\pi \leq \omega \leq \pi$, which is required to impart the correct interpretation to discrete frequency$^1$. To correct this problem, define

$$x \mod 2\pi \triangleq x - 2\pi \left\lfloor \frac{x}{2\pi} + \frac{1}{2} \right\rfloor$$

where $\lfloor x \rfloor$ is the floor function, rounding its argument down to the largest integer, no larger than $x$. Note that $-\pi \leq x \mod 2\pi < \pi$. Now, since $x \mod 2\pi$ differs from $x$ by a multiple of $2\pi$, the mathematical equivalence in equation (5) also gives

$$X[k] = \left( \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \right) \bigg|_{\omega = (\frac{2\pi}{N}k) \mod 2\pi} = \hat{x}(\omega)|_{\omega = (\frac{2\pi}{N}k) \mod 2\pi} \quad (6)$$

The fact that the DFT is a sampled version of the DTFT and that the DFT is invertible means that this sampled representation must contain all the information in the spectrally continuous DTFT, provided of course, that the signal is zero outside the region over which the DFT is taken. In the general case, where $x[n]$ has infinite support, we can still sample the spectrally continuous DTFT $\hat{x}(\omega)$, and associate these samples with a hypothetical DFT, defined by $X'[k] = \hat{x}(\omega)|_{\omega = (\frac{2\pi}{N}k) \mod 2\pi}$. In this case, however, the inverse DFT of $X'[k]$ is given by

$$x'[n] = \sum_{m} x[n + mN], \quad 0 \leq n < N$$

This is a time-domain aliasing relationship. It is the analog of frequency aliasing which occurs when a signal is sampled too coarsely in the time domain; in this case, the aliasing occurs when a signal is sampled too coarsely in the frequency domain. Time-domain aliasing can easily occur if we are not careful when using the DFT (usually through an FFT implementation) to implement filtering operations, as we shall see shortly.

### 2.3 Filtering with the DFT

#### 2.3.1 Causal Filters

The DFT provides an efficient tool for implementing FIR digital filters. This fact is a direct consequence of the relationship between the DFT and the DTFT. Let $h[n]$ denote the impulse response of a causal FIR filter with region of support

$$\mathcal{R}_h = [0, M]$$

$^1$The interpretation of frequency in the discrete domain derives from the fact that the FT and the DTFT are identical for Nyquist band-limited signals. Although some people prefer to define the DTFT as having a periodically repeating spectrum, this confuses the relationship between discrete and continuous signals – it requires the introduction of impulse sequences and other mathematical anomalies that are best avoided.
We would like to filter an input signal $x[n]$ to obtain

$$y[n] = \sum_{m=0}^{M} h[m]x[n-m]$$

To begin, we shall suppose that the input signal $x[n]$ also has finite support with $N_x$ samples, given by $0 \leq n < N_x$. It follows that, after filtering, the output signal $y[n]$ will be zero outside the region $0 \leq n < N_x + M$. Thus, selecting

$$N = N_x + M$$

we see that both $x[n]$ and $y[n]$ are zero outside the region $0 \leq n < N$, meaning that their $N$-point DFT's are sampled copies of their respective DTFT's. The DFT/DTFT equivalence relationship of equation (6) may then be invoked, yielding

$$Y[k] = \hat{y}(\omega)|_{\omega=\left(\frac{2\pi}{N}k\right) \text{mod } 2\pi}$$

$$= \left[ \hat{x}(\omega) \hat{h}(\omega) \right]|_{\omega=\left(\frac{2\pi}{N}k\right) \text{mod } 2\pi}$$

$$= X[k] \times \left[ \hat{h}(\omega) \right]|_{\omega=\left(\frac{2\pi}{N}k\right) \text{mod } 2\pi}$$

$$= X[k] H[k], \quad 0 \leq k < N$$

Notice that we have not appealed to any specific properties of the DFT itself, other than its relationship to the DTFT. It is the DTFT’s convolution theorem which is at work here. Later, in Section 2.4, we mention a separate, related convolution theorem for the DFT which applies to “circular” convolution; however, it is important to distinguish between circular convolution and true convolution – the latter is usually the goal in filtering applications.

### 2.3.2 Non-Causal Filters

Suppose that the region of support of the FIR filter $h[n]$ is given by

$$\mathcal{R}_h = [-N_h, P_h]$$

(7)

where $N_h$ and $P_h$ are arbitrary integers, to be interpreted as the negative and positive extents of the impulse response $h[n]$. In this case, the easiest way to apply the DFT/DFTF equivalence relationship is to insist that the $N_x$-sample input signal $x[n]$ is zero outside the half-open interval,

$$\mathcal{R}_x = [N_h, N_x + N_h).$$

After all, we are free to adopt any convention we like regarding the index of the first non-zero sample in the input sequence. It follows that the filtered output is supported on the half-open interval

$$\mathcal{R}_y = [0, N_x + N_h + P_h)$$
Thus, so long as we choose \( N \geq N_x + N_h + P_h \), we know that the \( N \)-point DFT’s satisfy \( Y[k] = X[k] H[k] \), meaning that the filtering operation can be carried out by multiplication in the DFT domain.

It is worth remembering that \( H[k] \) here is defined to be equal to the true filter frequency response at \( \omega = (2\pi k/N) \mod 2\pi \), as opposed to the DFT of \( h[n] \). We could alternatively compute \( H[k] \) as the \( N \)-point DFT of a sequence \( \bar{h}[n] \), where

\[
\bar{h}[n] = \begin{cases} 
    h[n] & n \geq 0 \\
    \bar{h}[N + n] & n < 0 
\end{cases},
\]

but there is no real value in doing this; we might as well directly evaluate \( \hat{h}(\omega) \) at the locations \( \omega = (2\pi k/N) \mod 2\pi \) when pre-computing the \( H[k] \) coefficients.

### 2.3.3 Overlap-Add-Save Method

We have shown that true filtering of a finite length sequence \( x[n] \), by a finite support filter \( h[n] \), is equivalent to multiplication of their \( N \)-point DFT’s, so long as \( N \) is at least as large as the support of the filtered output signal. In Section 2.5, we will demonstrate a fast implementation for the DFT (known as the FFT) which allows an \( N \)-point DFT to be computed using only \( N \log_2 N \) real-valued multiplications. These two observations are the basis of a number of related strategies for implementing large FIR filters efficiently.

The DFT size \( N \) is generally limited by practical considerations such as delay, memory and (to a lesser extent) computation, while the input signals of interest are usually much larger, potentially with unbounded length. In this section, we describe the overlap-add-save method for filtering an unbounded input sequence \( x[n] \), using bounded DFT operations.

Let \( M = N_h + P_h \) denote the order of the filter \( h[n] \), where \( N_h \) and \( P_h \) are the negative and positive extents from equation (7). The DFT size \( N \) is fixed, and the input signal is broken into finite length segments \( x_p[n] \), each with \( N_x = N - M \) samples. Specifically,

\[
x_p[n] = \begin{cases} 
    x[n + pN_x] & 0 \leq n < N_x \\
    0 & \text{otherwise}
\end{cases}
\]

Evidently, \( x[n] \) is the sum of its translated segments,

\[
x[n] = \sum_p x_p[n - pN_x]
\]

This segmented representation of \( x[n] \) is illustrated in Figure 1.

From our earlier discussion, we know that each segment \( x_p[n] \) may be filtered using the \( N \)-point DFT. Specifically, it is sufficient to first shift the samples of \( x_p[n] \) by \( N_h \), so that the shifted signal \( \bar{x}_p[n] = x_p[n - N_h] \) has region of support

\[
\mathcal{R}_{x_p} = [N_h, N_h + N_x),
\]
Figure 1: Segmentation of a signal \( x[n] \) into segments, \( x_p[n] \), each of length \( N_x \).

then filter in the DFT domain, yielding a shifted output \( \tilde{y}_p[n] \) with

\[
\tilde{Y}_p[k] = \tilde{X}_p[k] H[k], \quad 0 \leq k < N
\]

From equation (8), together with linearity and time invariance of filtering, we see that the complete filtered signal may be formed as a sum of the translated, filtered segments as

\[
y[n] = \sum_p y_p[n - pN_x] = \sum_p \tilde{y}_p[n - pN_x + N_h]
\]

Figure 2 illustrates the complete process, whereby each shifted input segment \( \tilde{x}_p \) expands to fill the DFT sample support \([0, N]\) upon filtering, after which overlapping portions (shown with shading) are accumulated to form a shifted (by \( N_h \)) copy of the filter output \( \tilde{y}[n] \).

The complete filtering procedure may be summarized as follows:

1. Determine the segment length \( N_x \), from \( N_x = N - M \).
2. For each input segment \( x_p[n] \),
   
   (a) Shift \( x_p \) to the right by \( N_h \) so that the first \( N_h \) and last \( P_h \) entries in an \( N \)-point holding buffer are 0.
   
   (b) Compute the \( N \)-point DFT of the shifted sequence, \( \tilde{X}_p[k] \).
   
   (c) Find \( \tilde{Y}_p[k] \) from \( \tilde{Y}_p[k] = \tilde{X}_p[k] \times H[k] \). The factors \( H[k] \) are fixed for all segments and generally computed ahead of time.
   
   (d) Find \( \tilde{y}_p[n] \) by taking the inverse DFT of \( \tilde{Y}_p[k] \).
   
   (e) Add the last \( M = N_h + P_h \) samples of \( \tilde{y}_{p-1}[n] \) (saved in the previous segment’s “save” step) to the first \( M \) samples of \( y_p[n] \), yielding the modified filtered output, \( \tilde{y}_p[n] \).
   
   (f) Output the first \( N_x \) samples of \( \tilde{y}_p[n] \), saving the remaining \( M \) samples in a temporary buffer for use in step (2.e) when processing the next segment. These \( N_x \) output samples from segment \( p \) correspond to sample locations \( n = N_xp - N_h \) through \( N_xp - N_h + N_x - 1 \) in the filtered output sequence \( y[n] \).
Figure 2: Filtering in the DFT domain, with a filter impulse response $h[n]$, supported on $R_h = [-N_h, P_h]$. The figure illustrates segmentation of the original signal $x[n]$ into length $N_x$ segments, followed by shifting of the segments by $N_h$, filtering with $N$-point DFT’s, in which $N = N_x + N_h + P_h$, and finally adding the overlapped segments to produce a shifted copy of the filter output.
Evidently, the algorithm calls for an $M$-sample save buffer, in addition to an $N$-sample buffer for performing the various DFT operations.

At this point, it is worth pointing out that the shifting by $N_b$, which we have used in the development above, is not strictly necessary. We could, for example, have shifted the filter instead, so that $N_b$ becomes 0. In any event, shifting here is really only a notional device to keep the connection between true convolution and convolution in the DFT domain as close as possible. The shifting processes do not actually incur any computational or memory access costs here.

To understand the computational advantages of DFT-based filtering, observe that the algorithm generates $N_x$ new samples after processing each segment. To generate these samples, an $N$-point DFT and an $N$-point IDFT must be performed, having a combined cost of $2N \log_2 N$ real-valued multiplications (see Section 2.5). Multiplication of $\tilde{X}_p[k]$ by $H[k]$ need only be performed over half of the DFT coefficients, due to conjugate symmetry, $Y_p[k] = Y_p^*[N - k]$, for a total cost of $N/2$ complex-valued multiplications, or $2N$ real-valued multiplications. The total number of multiplications$^2$ per output sample is then

$$
\frac{2N \log_2 2N}{N_x} = \frac{2N}{N - M} \log_2 2N
$$

**Example 1** A 1024-point DFT/IDFT processor is available for use in a filtering application, involving an order $M = 128$ FIR filter. The total number of multiplications required per output sample is

$$
\frac{2048}{1024 - 128} \log_2 2048 = 25.14
$$

By contrast, direct application of the filter requires 128 multiplications per output sample.

Some consideration should confirm the fact that DFT-based techniques are most interesting when working with very long filters. As an example, to simulate the acoustic environment of a typical music hall might require filters with many thousands of taps. In this case, the potential computational savings from a DFT-based implementation can be enormous. As we shall see, similar considerations apply in multiple dimensions.

### 2.4 Some Properties of the DFT

**Conjugate Symmetry:** When the input sequence $x[n]$ is real-valued, as is almost invariably the case for signal processing applications,

$$
X[k] = X^*[N - k]
$$

where the superscript $^*$ denotes complex conjugation as always.

---

$^2$A similar number of additions is required, but multiplications are generally more costly than additions.
Circular Convolution: Recall that the DFT is defined only on $0 \leq n < N$. The circular convolution of two sequences, $x[n]$ and $h[n]$, on this interval is defined by

$$y[n] = \sum_{k=0}^{N-1} x[k] h[(n - k) \mod N]$$

$$= \sum_{k=0}^{N-1} h[k] x[(n - k) \mod N]$$

In the DFT domain, circular convolution is always equivalent to multiplication, i.e.

$$Y[k] = X[k] \cdot H[k]$$

but the conditions under which circular convolution and true convolution are identical are those discussed in Section 2.3.

Dual of Convolution: Suppose we multiply two sequences $x[n]$ and $y[n]$ in the time domain to obtain $u[n] = x[n] \cdot y[n]$. In the DFT domain, this operation is equivalent to circular convolution. Specifically,

$$U[k] = \frac{1}{N} \sum_{p=0}^{N-1} X[p] Y[(k - p) \mod N]$$

Energy: Parseval’s relation states, essentially, that the energy in the time domain is equal to the energy in the Fourier domain. For the DFT, Parseval’s relation is

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

This may be derived easily from the normalized vector space interpretation of the DFT and its inverse in equations (3) and (4).

Extended Parseval Relationship: An extended form of Parseval’s relationship, for two arbitrary sequences, $x[n]$ and $y[n]$, states that

$$\sum_{n=0}^{N-1} x[n] y^*[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] Y^*[k]$$

2.5 Fast Algorithms for the DFT

Although fast DFT algorithms can be developed for arbitrary $N$, the savings are greatest and the development simplest when $N$ is a power of 2. For this reason, let $N = 2^r$ for some $r$. The basic FFT (Fast Fourier Transform) algorithm is
based on the observation that \( X[k] \) may be written as

\[
X[k] = \sum_{n=0}^{2^r-1} x[n] \exp\left(-j2\pi kn\right)
\]

\[
= \sum_{n=0}^{2^r-1} x[2n] \exp\left(-j2\pi kn\right)
\]

\[
+ \sum_{n=0}^{2^r-1} x[2n+1] \exp\left(-j2\pi kn\right) \cdot \exp\left(-j2\pi k\right)
\]

Now let \( x_e[n] = x[2n] \) denote the even sub-sequence of \( x[n] \) and let \( x_o[n] = x[2n+1] \) denote the odd sub-sequence. The DFT of each of these sub-sequences, \( X_e[k] \) and \( X_o[k] \), is defined for \( 0 \leq k < 2^{r-1} \). The above formula becomes

\[
X[k] = \begin{cases} 
X_e[k] + W_{2^r}^{-k}X_o[k] & \text{for } 0 \leq k < 2^{r-1}, \\
X_e[k - 2^{r-1}] + W_{2^r}^{-k}X_o[k - 2^{r-1}] & \text{for } 2^{r-1} \leq k < 2^r.
\end{cases}
\]

where we have used the notation,

\[
W_N = e^{j \frac{2\pi}{N}}
\]

introduced previously.

Finally, noting that \( W_{2^r}^k = -W_{2^{r-1}+k} \), we may express the \( 2^r \)-point DFT of \( x[n] \) in terms of the \( 2^{r-1} \)-point DFT’s of its even and odd sub-sequences as follows

\[
\begin{pmatrix} 
X[k] \\
X[k + 2^{r-1}]
\end{pmatrix} = \begin{pmatrix} 
X_e[k] + W_{2^r}^{-k}X_o[k] \\
X_e[k - 2^{r-1}] + W_{2^r}^{-k}X_o[k - 2^{r-1}]
\end{pmatrix}, \quad 0 \leq k < 2^{r-1}
\]

(9)

Now suppose that a \( 2^r \)-point FFT requires \( C_r \) operations per sample, or a total of \( 2^r C_r \) operations. The following recursive description of the algorithm allows us to compute its complexity in a straightforward manner:

1. Compute \( X_e[k] \) and \( X_o[k] \) for \( 0 \leq k < 2^{r-1} \). The number of operations required for this step is \( 2^r C_{r-1} \).

2. Multiply \( X_o[k] \) by \( W_{2^r}^{-k} \) for \( 0 \leq k < 2^{r-1} \), to obtain \( X'_o[k] \). We assume that the factors, \( W_{2^r}^{-k} \), have been computed and stored in advance, which is reasonable because they do not depend upon the signal, \( x[n] \). This step requires \( 2^{r-1} \) complex-valued multiplications.

3. For \( 0 \leq k < 2^{r-1} \) compute \( X[k] = X_e[k] + X'_o[k] \) and \( X[k + 2^{r-1}] = X_e[k] - X'_o[k] \). This step requires \( 2^r \) complex-valued additions/subtractions.\(^3\)

\(^3\)Since the complexity of additions is identical to the complexity of subtractions, we lump them together and generally refer to both as additions.
In terms of complex-valued multiplications, the operation count thus satisfies

\[
2^r C_r = 2^r C_{r-1} + \frac{1}{2} 2^r
\]

\[
= 2^r C_{r-2} + \frac{1}{2} 2^r + \frac{1}{2} 2^r
\]

\[
= \cdots
\]

\[
= \frac{r}{2} 2^r
\]

The number of complex-valued additions is twice as large, i.e. \( r 2^r \). These complexities are usually expressed in terms of the value of \( N = 2^r \). Thus, the number of complex-valued multiplications required to compute the \( N \)-point DFT using this simple FFT algorithm is \( \frac{N}{2} \log_2(N) \), while the number of complex-valued additions is \( N \log_2(N) \). Compare this with the direct computation of the FFT which requires \( N^2 \) complex-valued multiplications and additions.

### 2.6 Real-Valued Sequences

In the above formulation, the input sequence, its DFT and all numerical operations are complex-valued. In practice, we normally work with real-valued input sequences \( x[n] \), whose DFT satisfies the conjugate symmetry property, \( X[k] = X^*[N-k] \). We can exploit this symmetry to reduce the overall complexity.

One simple way to do this is to observe that at each stage of the FFT algorithm we need only compute \( X[k] \) over the range \( 0 \leq k < \frac{N}{2} \), using values of \( X_e[k] \) and \( X_o[k] \) in the range \( 0 \leq k < \frac{N}{4} \). The relevant update relationships are readily derived from equation (9) as

\[
\begin{pmatrix}
    X[k] \\
    X[2^{r-1} - k]
\end{pmatrix}
= \begin{pmatrix}
    X_e[k] + W_{2^r}^{-k} X_o[k] \\
    X_e^*[k] - W_{2^r}^k X_o^*[k]
\end{pmatrix}
\]

\[
= \begin{pmatrix}
    X_e[k] + W_{2^r}^{-k} X_o[k] \\
    X_e^*[k] - (W_{2^r}^{-k} X_o[k])^*
\end{pmatrix}, \quad 0 \leq k < 2^{r-2}
\]

The complexity of these calculations is identical to that of the full complex-valued calculations described previously, except that there are only half as many of them. Finally, observing that a complex-valued multiplication requires four real-valued multiplications and a complex-valued addition requires two real-valued additions, we find that the complexity of an \( N \)-point real-valued FFT is

\[
N \log_2 N \text{ real-valued multiplications, and}
\]

\[
N \log_2 N \text{ real-valued additions}
\]

It is worth noting that there are many variations on the basic FFT algorithm, as described here. In fact, there is at least one entire book, which deals exclusively with fast Fourier transform algorithms. Common to all these algorithms is the fact that the complexity of an \( N \)-point DFT has order \( N \log_2 N \).
3 The DFT in Multiple Dimensions

3.1 Definition
Recall that the Discrete Space Fourier Transform (DSFT) is defined on \( m \)-dimensional sequences of infinite extent according to

\[
\hat{x}(\omega) = \sum_n x[n] e^{-j n^t \omega}, \quad \text{for } \omega \in [-\pi, \pi]^m
\]

(10)

with the inverse DSFT given by

\[
x[n] = \frac{1}{(2\pi)^m} \int_{-\pi}^{\pi} \cdots \int_{-\pi}^{\pi} \hat{x}(\omega) e^{j n^t \omega} d\omega
\]

(11)

The \( m \)-dimensional DFT is defined only for sequences \( x[n] \) of finite support. For simplicity, we will take \( x[n] \) to be zero outside the hypercube

\[
\mathcal{R}_N = [0, N]^m
\]

\( \mathcal{R}_N \) is the set of all coordinates \( n \equiv (n_1, \ldots, n_m) \), such that

\[
0 \leq n_i < N, \text{ for } i = 1, 2, \ldots, m.
\]

With this convention, the \( N \)-point \( m \)-dimensional DFT and its inverse are given by the following equations:

\[
X[k] = \sum_{n \in \mathcal{R}_N} x[n] e^{-j \frac{2\pi n^t k}{N}}, \quad k \in \mathcal{R}_N
\]

(12)

\[
x[n] = \frac{1}{N^m} \sum_{k \in \mathcal{R}_N} X[k] e^{-j \frac{2\pi n^t k}{N}}, \quad n \in \mathcal{R}_N
\]

(13)

3.2 Relationship Between DFT and DSFT
As in one dimension, the DFT and DSFT are equivalent, so long as \( x[n] \) is zero outside \( \mathcal{R}_N \). Combining equations (10) and (12), it can be seen that the \( N \)-point DFT of an \( m \)-dimensional sequence which is zero outside \( \mathcal{R}_N \) may be written as

\[
X[k] = \sum_n x[n] e^{-j \frac{2\pi n^t k}{N}} = \hat{x}(\omega)|_{\omega = (\frac{2\pi k}{N}) \mod 2\pi}
\]

where the \( u \mod 2\pi \) denotes the vector

\[
u \mod 2\pi = \begin{pmatrix} u_1 \mod 2\pi \\ \vdots \\ u_m \mod 2\pi \end{pmatrix},
\]

whose components are individually wrapped back into the Nyquist box, as required.
3.3 Multi-Dimensional Filtering with the DFT

As in the one-dimensional case, we can implement convolution of a finite support sequence $x[n]$ by an FIR impulse response $h[n]$ in the $N$-point DFT domain, so long as both $x[n]$ and the filtered output $y[n]$ are zero outside $\mathcal{R}_N = [0, N)^m$.

Consider, for example, a PSF $h[n]$ with region of support $\mathcal{R}_h = [-N_h, N_h]^m$ and suppose that $x[n]$ is zero outside $\mathcal{R}_x = [N_h, N_x + N_h]^m$.

Then the filtered output $y[n]$ is zero outside $\mathcal{R}_y = [0, N_x + 2N_h]^m$, and so for $N \geq N_x + 2N_h$, we have

$$Y[k] = X[k] \cdot H[k], \quad k \in \mathcal{R}_N$$

where $H[k]$ is defined by

$$H[k] = \hat{h}(\omega) \bigg|_{\omega = (\frac{2\pi k}{N}) \mod 2\pi},$$

(14)

Filtering can thus be performed by multiplication in the DFT domain.

As in the one-dimensional case, we can filter a much larger signal (image, video, etc.) using a comparatively small DFT, by representing the input sequence as a sum of disjoint segments (blocks for $m=2$, cubes for $m=3$, etc.), i.e.,

$$x[n] = \sum_p x_p[n - N_x p]$$

where $N_x$ is chosen to be significantly larger than $2N_h$. Each segment is individually convolved by $h[n]$ via the following steps:

1. Situate the segment $x_p$ within a block of $\mathcal{R}_N$ samples, by padding it with zeros, extending at least $N_h$ samples around all boundaries. Call this extended segment $\tilde{x}_p$.

2. Take the $m$-dimensional DFT of $\tilde{x}_p$ and form the DFT of the corresponding output segment $\tilde{y}_p$ as

$$\tilde{Y}_p[k] = \tilde{X}_p[k] \cdot H[k]$$

where the $H[k]$ coefficients have been precomputed using equation (14).

3. Take the inverse DFT of $\tilde{Y}_p[k]$.

4. Correctly align and add up the overlapping portions of the filtered segments $\tilde{y}_p[n]$, yielding.
3.4 Some Properties of the Multi-Dimensional DFT

Conjugate Symmetry: When the input sequence \( x[n] \) is real-valued, as is usually the case,

\[ X[k] = X^*[N - k] \]

where \( N - k \) denotes the vector

\[ N - k = \begin{pmatrix} N - k_1 \\ \vdots \\ N - k_m \end{pmatrix} \]

Separability: See the following sub-section.

Circular Convolution: Recall that our \( m \)-dimensional \( N \)-point DFT is defined only on \( \mathcal{R}_N \). The circular convolution of two sequences, \( x[n] \) and \( h[n] \), on this interval is defined by

\[
y[n] = \sum_{k \in \mathcal{R}_N} x[k] h[(n - k) \mod N]
\]

In the DFT domain, circular convolution is always equivalent to multiplication, i.e.

\[ Y[k] = X[k] \cdot H[k]. \]

Dual of Convolution: Suppose we multiply two sequences \( x[n] \) and \( y[n] \) in the time domain to obtain \( u[n] = x[n] \cdot y[n] \). In the DFT domain, this operation is equivalent to circular convolution. Specifically,

\[
U[k] = \frac{1}{N^m} \sum_{p \in \mathcal{R}_N} X[p] Y[(k - p) \mod N] \tag{15}
\]

Energy: Parseval’s relation states, essentially, that the energy in the space domain is equal to the energy in the Fourier domain. For the DFT, Parseval’s relation is

\[
\sum_{n \in \mathcal{R}_N} |x[n]|^2 = \frac{1}{N^m} \sum_{k \in \mathcal{R}_N} |X[k]|^2
\]

Extended Parseval Relationship: An extended form of Parseval’s relationship, for two arbitrary sequences, \( x[n] \) and \( y[n] \), states that

\[
\sum_{n \in \mathcal{R}_N} x[n] y^*[n] = \frac{1}{N^m} \sum_{k \in \mathcal{R}_N} X[k] Y^*[k]
\]
3.5 Fast Implementations and the Multi-Dimensional FFT

From equations (12) and (13), it can be seen that the multi-dimensional DFT and inverse DFT are separable operators. For example, we can write

\[ X[k] = \sum_{n \in \mathbb{R}^N} x[n] e^{-j \frac{2\pi n' k}{N}} = \sum_{n_1=0}^{N} e^{-j \frac{2\pi n_1 k_1}{N}} \cdots \sum_{n_m=0}^{N} e^{-j \frac{2\pi n_m k_m}{N}} x[n_1, \ldots, n_m] \]  

(16)

In two dimensions, for example, it is sufficient to first take the 1D DFT of each row in an image, followed by the 1D DFT of each column of the result. Alternatively, we can start with the columns and then process rows.

For a complex-valued input sequence, we have seen that the complexity of an \( N \)-point 1D DFT is \( N^2 \log_2 N \) complex-valued multiplies and \( N \log_2 N \) complex-valued additions. This means that the per-sample processing cost of applying the 1D FFT along any dimension of \( x[n] \) is \( 2 \log_2 N \) real-valued multiplications and additions. Considering that there are \( m \) dimensions, the total cost is \( 2m \log_2 N \) multiplications and additions per sample, or \( 2N^m \log_2(N^m) \) multiplications and additions in total, for the entire sequence.

For a real-valued input sequence, we have seen that conjugate symmetry can be exploited to reduce the complexity of the 1D FFT by a factor of 2. This general observation is still true in multiple dimensions, but we need to be careful when extending the approach described in Section 2.6; the point to note is that after the initial application of a 1D FFT along the \( n_m \) dimension, following equation (16), the result is now complex-valued. In order to realize the factor of 2 saving overall, we proceed as follows:

1. Start by taking the 1D DFT of each real-valued row (or whatever the \( n_m \) dimension may be), at a total cost of \( N^m \log_2 N \) multiplications and additions.

2. Next, take the 1D DFT of just the first half of the columns (or whatever the \( n_m - 1 \) dimension may be). These are all complex-valued, but we need only process half the columns, for a total cost of \( N^m \log_2 N \) multiplications and additions. In this way, we compute the 2D DFT for exactly half the frequency coefficients. The remaining half can be recovered by conjugate symmetry, recognizing that \( X[k] = X^*[N - k] \).

3. If there are additional dimensions, we proceed in the same way, extending the 2D DFT results to 3D DFT coefficients over half the frequency coefficients and filling in the rest by conjugate symmetry.
In this way, the overall cost of performing an $N$-point $m$-dimensional FFT on a real-valued signal is $N^m m \log_2 N = N^m \log_2 (N^m)$ real-valued multiplications, and the same number of additions.

It is perhaps worth noting that there are some less obvious algorithms for computing multi-dimensional FFT’s which have somewhat lower complexity (by a small factor). One such algorithm, due to Nussbaum and Quandalle [1], can reduce the total number of multiplications by a factor of approximately $m$, although the number of additions is not reduced.

4 Use of the DFT in “Pattern Matching”

In this section, we consider techniques for locating features in an image, based upon intensity patterns. Similar methods apply in one and three dimensional media sources. For example, we may use these techniques to search for patterns in an auditory waveform, or patterns in a 3D medical image volume.

4.1 Correlation and Matched Filtering

Let $x[n]$ be the image and let $y[n]$ be an intensity pattern which we wish to match within the image. We may think of $x[n]$ and $y[n]$ as having infinite extent, but equal to zero outside their respective regions of support. Although the pattern $y[n]$ is itself an image, it generally has a much smaller region of support than the full image, $x[n]$. For the sake of simplicity, we shall assign a square region of support to the pattern and the image, with

$$y[n] = 0 \quad \text{except in the range } -M \leq n_1, n_2 \leq M$$

and

$$x[n] = 0, \quad \text{except in the range } 0 \leq n_1, n_2 < N$$

where $N \gg M$.

One reasonable way to search for occurrences of the pattern within the image is to measure the mean square distance,

$$E[p] = \sum_n (x[n] - y[n - p])^2 \quad \text{(17)}$$

between $x[n]$ and shifted copies of the pattern, $y[n - p]$, where $p$ is the shift. The shift, $p_0$, at which $E[p]$ achieves its minimum value, may be interpreted as the location in the image which best matches the pattern.

Some simple manipulation of equation (17) yields

$$E[p] = \sum_n (x[n] - y[n - p])^2$$

$$= \sum_n x^2[n] + \sum_n y^2[n - p] - 2\sum_n x[n]y[n - p]$$
The first two terms on the right hand side have no dependence upon \( p \), and so \( \mathcal{E} [p] \) is minimized by maximizing

\[
r [p] = \sum_n x [n] y [n - p]
\]

(18)

The process of forming the image \( r [p] \), is known as correlation.

### 4.1.1 Matched Filtering and the FFT

Equation 18 may be rewritten as

\[
r [p] = \sum_n x [n] \hat{y} [p - n] = (x \ast \hat{y}) [p]
\]

where \( \hat{y} [n] = y [-n] \) is the mirror image of the pattern, \( y [n] \). Thus, correlation may equivalently be viewed as filtering the image with an FIR filter having region of support \( \mathcal{R}_y = [-M, M]^2 \). This “filter” is known as the “matched filter.” The maxima of the matched filter are the most likely locations of the pattern within the image. Matched filtering is known to be the optimal strategy for detecting a signal embedded in white noise. Unfortunately, in our case the signal is embedded within the image, which looks nothing like white noise.

The matched filter cannot be expected to possess any special structure, such as separability or symmetry, so the direct cost of evaluating \( r [p] \) at each location \( p \) is \((2M + 1)^2\) multiplications and additions. For large patterns, this can be prohibitively expensive. Fortunately, however, the matched filter has finite extent so we can use FFT techniques to dramatically reduce the filtering complexity. When applying these techniques you must be especially careful to provide enough zero padding to ensure that multiplication in the DFT domain is exactly equivalent to true convolution in the image domain. As seen in Section 3.3, with sufficient zero padding, the DFTs of \( r [p] \), \( x [n] \) and \( y [n] \) are related through

\[
R [k] = X [k] Y^* [k].
\]

Moreover, if the pattern is indeed much smaller than the image, the overlap-add-save method described at the end of Section 3.3 may be applied also to the correlation problem: i.e., divide the image into disjoint blocks; take the correlation of each block in the DFT domain; add the overlapping portions of the filtered blocks; and find the local maxima of the result.

### 4.1.2 A Simple Example

Figure 3 provides a simple example of pattern matching by correlation. The original image is shown in the upper left hand corner of the figure, while the pattern of interest is shown to the right. Note that the pattern has much smaller region of support than the image itself. The correlation image is shown at the lower left, with its two strongest local maxima highlighted at the lower right. Evidently, if the pattern can occur multiple times within the image, we must
develop some criterion for rejecting all but the strongest local maxima. In this highly controlled case, a simple threshold is sufficient.

4.2 Problems with Correlation

As mentioned above, correlation (or matched filtering) is known to be the optimal strategy for detecting a signal in white noise, but the image is not well modeled by white noise. To explore the difficulties created by this modeling mismatch, we begin by defining $N_y$ to be the geometric region (or neighbourhood) over which the pattern $y[n]$ is considered to exist. This may or may not be the same as the region of support $R_y$, since $y[n] = 0$ might be a meaningful (black) sample in the pattern. In general then, we can expect that $N_y \supseteq R_y$.

Now observe that the pattern we are trying to match generally has non-zero mean, so that we may write

$$y[n] = \begin{cases} 
\mu_y + y'[n], & n \in N_y \\
0, & n \notin N_y 
\end{cases}$$

where $\mu_y > 0$ is the non-zero mean intensity over the pattern, and $y'[n]$ represents fluctuations about this mean.

We have seen that the location $p$, which minimizes the squared error in equation (17) is also the location which maximizes the correlation,

$$r[p] = \langle x, y_p \rangle = (x \ast \tilde{y})(p).$$
However, because $\mu_y > 0$, if we add some constant offset $A$ to the image samples in the neighbourhood of $p$, we will increase the value of $r(p)$ by the amount $A \cdot \|\mathcal{N}_y\| \cdot \mu_y$. This observation tells us that brighter areas of the image will produce larger values for the correlation $r[p]$, so that these are more likely to be selected as the best matching locations.

This conclusion may at first seem quite counter-intuitive. After all, maximizing the correlation is the same as minimizing the total squared error between $x[n]$ and $y[n]$, so it seems odd that adding to the brightness of some region of the image would improve the match in that region. Of course, it does not actually improve the match; rather, it makes all other candidate locations (those which have not been made brighter) much less attractive. The reason for this is best understood by considering a slightly different formulation of the matching problem, in which we seek the location $p$ which minimizes the local squared error

$$O[p] = \sum_{n \in \mathcal{N}_y + p} (x[n] - y[n - p])^2.$$  \hspace{1cm} (19)

Here, $\mathcal{N}_y + p$ is a translated version of $\mathcal{N}_y$, shifted to the location $p$. Equation (19) considers the mismatch between $y[n - p]$ and $x[n]$ only over the region represented by the pattern – there is no zero padding of $y[n]$ out to an infinite sequence here. Now suppose the image contains an exact replica of the pattern for which we are searching, located at some position $p_0$. At this location, $O[p_0] = 0$, so it is clear that no adjustment of the image brightness can improve this match. In fact, adding a constant $A$ increases our local squared error measure by $A^2 \cdot \|\mathcal{N}_y\|$.

### 4.3 Pre-Filtered Correlation

The problems described above can be largely avoided by pre-processing. To understand this approach, let $h_{hp}[n]$ denote any high-pass filter, which we can apply to both the input and pattern images to obtain

$$x'[n] = (x * h_{hp})[n] \text{ and } y'[n] = (y * h_{hp})[n]$$

Now if $y[n]$ translated by $p$ is a good match for the corresponding region within $x[n]$, then $y'[n]$ translated by $p$ should be a good match for $x'[n]$. Thus, we can proceed as before, forming the correlation

$$r'[p] = \sum_n x'[n] y'[n - p] = (x' * y')[p]$$

The benefit of this pre-filtered approach is that it is now reasonable to think of $y'[n]$ as being extended infinitely with zeros, since it is a high-pass pattern. The content from $x'[n]$ that is not related to the pattern can also be expected to be uncorrelated with the pattern, since $x'[n]$ and $y'[n]$ now have average values of zero.
To efficiently compute \( r'[p] \), we observe firstly that

\[
\begin{align*}
  r'[p] &= (x * h_{hp}) * (\hat{y} * h_{hp}) [p] \\
  &= (x * h_{hp}) * (\hat{y} * h_{hp}) [p] \\
  &= x * \left( \frac{h_{hp} * \hat{h}_{hp}}{h_{hp}} \right) \hat{y} [p] \\
  &= x * (\hat{y} * h_{shp}) [p] = (x * \hat{y}'') [p]
\end{align*}
\]

That is, we only need to pre-filter the pattern \( y[n] \) using the symmetric high-pass filter \( h_{shp} = h_{hp} * \hat{h}_{hp} \), yielding \( y''[n] \), after which \( r'[p] \) can be computed by efficient FFT-based convolution of \( x[n] \) with the mirror image of the pre-filtered pattern, \( \hat{y}''[n] \).

It is instructive to observe that the symmetric high-pass filter \( h_{shp} \) has

\[
\hat{h}_{shp}(\omega) = \hat{h}_{hp}(\omega) \cdot \hat{h}_{hp}^*(\omega) = |\hat{h}_{hp}(\omega)|^2
\]

which is real-valued and non-negative everywhere. This pre-filter thus introduces no phase changes to any frequency component of the pattern, since phase changes correspond to shifts. Moreover, being non-negative everywhere means that no frequency components make negative contributions to the correlation, which would undermine the idea that the location \( p \) with maximum correlation is the best match. Any pre-filter with these two properties (real and non-negative in the Fourier domain) can be used as an effective pre-filter for the pattern in correlation-based matching techniques. Effectively, the pre-filter reweights the contributions of different frequency components in \( \hat{y}(\omega) \) to the matching objective \( r'[p] \).

### 4.4 Phase Correlation

A popular variation on the correlation method described above is known as “phase correlation.” In this case, the magnitude of the correlation DFT, \( R[k] \), is set to 1 at every frequency, \( k \), before inverting the DFT and finding the maxima of the resulting phase correlation image, \( r'[p] \).

To understand why this might be useful, consider the true DSFT of the signals \( r[p], x[n] \) and \( y[n] \). These satisfy

\[
\hat{r}(\omega) = \hat{x}(\omega) \hat{y}^*(\omega)
\]

Now define \( r'[p] \) to be the image whose true DSFT satisfies

\[
\hat{r}'(\omega) = \frac{\hat{r}(\omega)}{|\hat{r}(\omega)|} = \frac{\hat{x}(\omega) \hat{y}^*(\omega)}{|\hat{x}(\omega)| \cdot |\hat{y}(\omega)|}
\]
so we may interpret $r'[p]$ as the true correlation between two images, $x'[n]$ and $y'[n]$, whose DSFTs are

$$\hat{x}'(\omega) = \frac{\hat{x}(\omega)}{|\hat{x}(\omega)|} \quad \text{and} \quad \hat{y}'(\omega) = \frac{\hat{y}(\omega)}{|\hat{y}(\omega)|}$$

These images are obtained from $x[n]$ and $y[n]$ by pre-filtering them using zero-phase filters whose magnitude responses are $|\hat{x}(\omega)|^{-1}$ and $|\hat{y}(\omega)|^{-1}$, respectively.

In this way, phase correlation can be interpreted as the correlation of pre-filtered versions of the original image and the pattern. The pre-filters tend to emphasize frequencies which are under-represented in the image and the pattern, respectively. This is often useful, because direct correlation produces results which are dominated by the spatial frequency components which occur everywhere in the image and are hence of little value in localizing the feature of interest.

To understand this, note that images tend to have strongly low-pass power spectra, so that high frequency components have much less energy than low frequency components. This means that direct correlation is dominated by the lower frequency components, carrying information about the local intensity of the image. This, in turn means that the correlation will most likely be maximized by the region of the image whose average intensity is closest to that of the pattern, regardless of the more subtle spatial fluctuations in the pattern. By contrast, phase correlation dramatically boosts the influence of higher frequency components, reducing the impact of the dominant but uninformative low frequency information.

Apparent then, phase correlation is similar to pre-filtered correlation, as discussed above, except that the pre-filter which re-weights the contribution of different frequency components, is image and pattern dependent, rather than a fixed symmetric high-pass filter $h_{\text{shp}}[n]$.

Another way to appreciate the potential benefits of phase correlation is to observe that the pre-filtered image, $\hat{x}'(\omega)$ has a white power spectrum. Recall that matched filtering is best suited to locating signals of interest within white noise.

It is worth noting that the above arguments have been made in the context of the true DSFT of the various images. Whereas $x[n]$ and $y[n]$ both have finite support, neither $x'[n]$ nor $y'[n]$ generally have finite support. Thus, when the phase correlation image is computed as the inverse DFT of $\hat{R}[k] / |\hat{R}[k]|$, as is invariably the case, the result will contain spatial aliasing artefacts which may interfere with the location of the maxima. These effects can be reduced by the use of longer DFT's with substantial zero padding. Note that a data-independent pre-filter $h_{\text{shp}}$, as discussed earlier, does not have this difficulty.
4.5 Normalized Cross Correlation

We have seen that the direct correlation of \(x[n]\) and \(y[n]\) can be unduly influenced by low frequency components, especially the local mean. An alternate strategy is to directly factor out the influence of the local image intensity. The so-called normalized cross-correlation method does exactly this and generally yields more reliable results.

The normalized cross-correlation between \(x[n]\) and \(y[n]\), with shift \(p\), is defined by

\[
\bar{r}[p] = \frac{\sum_{n \in N_y+p} (x[n] - \mu_x[p]) (y[n-p] - \mu_y)}{\sqrt{\sum_{n \in N_y+p} (x[n] - \mu_x[p])^2 \cdot \sum_{n \in N_y} (y[n] - \mu_y)^2}}
\]

(20)

Note that the sums in the above equation all extend only over the region occupied by the translated pattern, \(N_y + p\). The expression in the numerator represents the correlation between two images, \(x'[n]\) and \(y'[n]\), obtained by subtracting the local mean from the original images. The local mean \(\mu_x[p]\) of image \(x[n]\) is given by

\[
\mu_x[p] = \frac{1}{\|N_y\|} \sum_{n \in N_y+p} x[n]
\]

while \(\mu_y\) is the mean of the pattern image, \(y[n]\).

The expressions in the denominator of equation (20) represent the root mean square deviation between \(x[n]\) and its local mean and between \(y[n]\) and its mean, measured over the region within which the image and pattern are being correlated. These may be interpreted as measures of the local contrast of the image and the pattern.

To gain a greater understanding of normalized cross correlation, consider the \(\|N_y\|\)-dimensional vectors \(x'_p\) and \(y'\), whose elements are the samples values of \(x'[n+p]\) and \(y'[n]\), respectively, as \(n\) ranges over the locations in \(N_y\). Note that \(x'_p\) is just the portion of \(x[n]\) which lies beneath the pattern when it is translated to location \(p\). In terms of these vectors, the normalized cross correlation can be expressed as

\[
\bar{r}[p] = \frac{\langle x'_p, y' \rangle}{\sqrt{\langle x'_p, x'_p \rangle} \cdot \sqrt{\langle y', y' \rangle}} = \frac{\langle x'_p, y' \rangle}{\|x'_p\| \cdot \|y'\|}
\]

According to the Cauchy-Schwarz inequality\(^4\), the ratio of an inner product to the product of the norms of the two vectors, must always lie in the range \(-1\) to \(1\). That is,

\[-1 \leq \bar{r}[p] \leq 1.
\]

Moreover, \(\bar{r}[p]\) attains its maximum possible value of 1 if and only if \(x'_p\) is a scalar multiple of the mean-removed pattern \(y'\). More generally, \(\bar{r}[p]\) measures the cosine of the angle formed between the two vectors (images) \(x'_p\) and

\(^4\)This is the most fundamental inequality which must be satisfied by any valid inner product.
y'. The normalized cross correlation measure thus deems the pattern to be best matched by those regions of the image whose variation about their mean intensity resembles that of the pattern, up to a scaling factor.

4.5.1 Efficient Implementation

Since we are only interested in finding the locations $p$, which maximize $\tilde{r} [p]$, the cross correlation expression can be simplified by dropping the fixed contrast of the pattern image – i.e., the last term in the denominator of equation (20). Also, we may work with a modified pattern image $y' [n]$, which has already had the mean value removed. In this way, it is sufficient to evaluate the expression

$$K (p) = \frac{\sum_{n \in N_y + p} (x [n] - \mu_x [p]) y' [n - p]}{\sqrt{\sum_{n \in N_y + p} (x [n] - \mu_x [p])^2}}$$

at each location $p$.

A final important simplification arises when we observe that

$$\sum_{n \in N_y + p} (x [n] - \mu_x [p]) y' [n - p] = \sum_{n \in N_y + p} x [n] y' [n - p] - \mu_x [p] \sum_{n \in N_y + p} y' [n - p]$$

$$= \sum_{n \in N_y + p} x [n] y' [n - p]$$

since $y' [n]$ has the mean removed. Our objective is thus to find the locations $p$, which maximize

$$K (p) = \frac{\sum_{n} x [n] y' [n - p]}{\sqrt{\sum_{n \in N_y + p} (x [n] - \mu_x [p])^2}}$$

where we have dropped the range on the summation in the numerator, taking $y' [n]$ to be zero outside the region $N_y$.

The above manipulations reveal the fact that so long as we are careful to always remove the mean from the pattern image, the normalized cross-correlation measure is equivalent to taking the direct correlation of the image and the pattern, but dividing the result by the local image contrast. The numerator may be computed efficiently using FFT techniques, while the denominator may be expressed as

$$\sqrt{\sum_{n \in N_y + p} (x [n] - \mu_x [p])^2} = \sqrt{\sum_{n \in N_y} x^2 [n + p] - \mu_x^2 [p]}$$

For rectangular pattern regions, $N_y$, moving average filters may be used to efficiently compute $\sum_{n \in N_y} x^2 [n + p]$ and $\mu_x [p] = \frac{1}{|N_y|} \sum_{n \in N_y} x [n + p]$, with constant complexity, independent of the size of the pattern.
5 Use of the DFT in Power Spectrum Estimation

5.1 Introduction to Power Spectrum Density (PSD)

Power spectrum density is a property of random processes, not of deterministic signals. Of course, we observe some signal $x[n]$, but the objective of power spectrum estimation is to estimate the statistical properties of an underlying random process $X[n]$, of which $x[n]$ is a specific realization. In many cases, $X[n]$ has infinite support, at least in one dimension (e.g., time for a video sequence), but we wish to estimate the power spectrum based upon a limited set of observations, $x[n]$.

While power spectrum estimation is not a central focus of the present course, it is important both in guiding the design of filters and also in the analysis of textures – a topic that we will encounter later. Most students, and many professional engineers, immediately reach for the DFT (or its fast FFT implementation) as a tool for estimating the power spectrum in a signal. Unfortunately, direct application of the DFT produces very poor estimates indeed, so it is important that you understand why the DFT by itself is not a good spectrum analyzer and what can be done to improve it.

5.1.1 PSD of a 1D Random Process

Although a lot could be said about the definition and interpretation of the power spectrum of a random process, this section must necessarily provide only the briefest of introductions. We begin with one-dimensional random processes $X[n]$. A random process is a sequence of random variables. By itself, this is not all that interesting, unless we add additional properties which connect the random variables with their neighbours in the sequence. A most important concept is that of stationarity.

One property of a stationary random process is that all of the random variables $X[n]$ have the same underlying probability distribution; certainly, they have a common mean

$$
\mu_X = E[X[n]] \quad \text{for all } n
$$

and a common variance

$$
\sigma_X^2 = E[(X[n] - \mu_X)^2] = E[X^2[n]] - \mu_X^2.
$$

However, stationarity is a much stronger property. For a process to be truly stationary, the joint probability distribution associated with every finite set of elements from $X[n]$ must be identical to the probability distribution of all translated versions of itself. In particular, every vector of the form

$$
X_p^m = (X[p], X[p+1], \ldots, X[p+m-1])^T
$$

has the same joint probability distribution as $X_{p+n}^m$, for all integers $m$, $p$ and $n$. 
A much weaker property, known as “Wide Sense Stationarity,” is often sufficient for signal processing applications. A Wide Sense Stationary (WSS) random process \( X[n] \) is one in which all random variables have the same mean \( \mu_X \) and the auto-correlation function is independent of \( n \). That is,

\[
R_X[p, n] = E[X[n] \cdot X[n-p]]
\]

depends only on the lag \( p \); we may thus write

\[
R_X[p] = E[X[n] \cdot X[n-p]],
\]
treating \( n \) as a dummy variable, or placeholder.

In one-dimensional audio signal processing, the mean \( \mu_X \) is often 0. For visual media, however, this is rarely the case. In the event that the mean is non-zero, a more meaningful statistic than auto-correlation is the auto-covariance

\[
C_X[p] = E[(X[n] - \mu_X) \cdot (X[n-p] - \mu_X)]
\]

\[
= E[X[n] \cdot X[n-p]] - \mu_X E[X[n]] - \mu_X E[X[n-p]] + \mu_X^2
\]

It is worth noting that the auto-covariance at lag 0 is the variance, i.e., \( C_X[0] = \sigma_X^2 \).

For WSS random processes, the Power Spectrum Density (PSD) may be defined either as the Fourier transform of the auto-correlation sequence \( R_X[p] \), or as the Fourier transform of the auto-covariance sequence \( C_X[p] \). The difference between these two definitions is that the former contains infinite power at DC \( (\omega = 0) \) whenever the random process has non-zero mean. This is inconvenient, and not all that informative. For this reason, we will define the PSD as

\[
\Gamma_X(\omega) = \mathcal{F}(C_X[m])
\]

\[
= \sum_{p=-\infty}^{\infty} C_X[m] e^{-j\omega p}, \quad \text{for } \omega \in [-\pi, \pi]
\]

Some interesting properties of \( \Gamma_X(\omega) \) are as follows:

**Non-negativity:** Reassuringly, the PSD is real-valued\(^5\) and strictly non-negative for all \( \omega \).

**Integration:** The variance \( \sigma_X^2 \) of any variable in the random process may be obtained simply by integrating (actually, averaging) the PSD over \([-\pi, \pi]\]. That is,

\[
\sigma_X^2 = C_X[0] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_X(\omega) \, d\omega
\]

(21)

All we are doing here is evaluating the inverse DFT of \( \Gamma_X(\omega) \) at \( p = 0 \). We normally interpret the variance of a random process as its power, although sometimes we think of \( R_X[0] = \sigma_X^2 + \mu_X^2 \) as the power instead. Of course, when the mean is zero, both interpretations are identical.

\(^5\)That \( \Gamma_X(\omega) \) is real-valued arises from the fact that \( C_X[m] = C_X[-m] \), which is obvious from the definition of \( C_X[m] \).
**Filtering:** Let \( Y[n] \) be the random process obtained by convolving \( X[n] \) with a deterministic filter impulse response \( h[n] \). That is,
\[
Y[n] = \sum_k h[k] X[n-k]
\]
What we mean by this is that every sequence \( y[n] \), which is a realization of \( Y[n] \), may be found by convolving a corresponding realization \( x[n] \) of \( X[n] \) by \( h[n] \). In any event, the power spectrum of \( Y[n] \) may be readily shown to satisfy
\[
\Gamma_Y(\omega) = |\hat{h}(\omega)|^2 \Gamma_X(\omega)
\]
Amongst other things, this allows us to compute the power (variance) of the output process from equation (21), obtaining
\[
\sigma_Y^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |\hat{h}(\omega)|^2 \Gamma_X(\omega) \, d\omega. \quad (22)
\]
Equation (22) is exactly what we would expect of a power density function. In particular, suppose \( h[n] \) is an ideal narrow band filter, with the property that
\[
\hat{h}(\omega) = \begin{cases} 
1, & \omega \in (\omega_0 - \delta, \omega_0 + \delta) \\
1, & \omega \in (-\omega_0 - \delta, -\omega_0 + \delta) \\
0, & \text{otherwise}
\end{cases}
\]
That is, \( h[n] \) passes signal content only within neighbourhoods of size \( 2\delta \), centered about each of \( \omega_0 \) and \( -\omega_0 \). For small enough \( \delta \), equation (22) tells us that the output power satisfies
\[
\sigma_Y^2 \approx \frac{2\delta}{\pi} \Gamma_X(\omega_0),
\]
which is (up to a constant scaling factor) the size of the neighbourhood, multiplied by the power density within the neighbourhood.

### 5.1.2 PSD for Multi-Dimensional Random Processes

A multi-dimensional random process \( X[n] \) is, again, just a collection of random variables. The properties of stationarity and wide sense stationary are the natural extensions of their one-dimensional counterparts. In particular, stationarity means that the joint distribution of any collection of random variables from the process is invariant to translation, while wide sense stationarity means that the mean and correlation/covariance statistics are invariant to translation. A WSS random process has mean, auto-correlation and auto-covariance sequences, defined by
\[
\mu_X = E[X[n]],
\]
\[
R_X[p] = E[X[n]X[n-p]] \quad \text{and}
\]
\[
C_X[p] = E[(X[n]-\mu_X)(X[n-p]-\mu_X)] = R_X[p]-\mu_X^2,
\]
all of which are independent of \( n \).

The PSD of a WSS process \( X[n] \) is again just the Fourier transform of the auto-covariance sequence,

\[
\Gamma_X(\omega) = \sum_{p} C_X[p] e^{-j\omega p}
\]

having the same properties as its one-dimensional counterpart.

### 5.1.3 Typical Power Spectra

If a discrete random process \( X[n] \) is obtained by sampling a Nyquist bandlimited continuous random process \( X(s) \), the PSD of the discrete and continuous random processes are identical. Such a bandlimited continuous random process may arise by the application of an anti-aliasing filter to a fundamental source process \( R(s) \) prior to sampling. If we were able to progressively increase the sampling density — e.g., by scaling \( R(s) \) prior to application of the filtering and sampling processes — we could expect that the discrete power spectrum \( \Gamma_X(\omega) \) would capture progressively higher and higher frequency content from the underlying continuous process. We could write this as

\[
\Gamma_X(\omega) = \Gamma_R(\alpha \cdot \omega), \quad \omega \in [-\pi, \pi]^m
\]

where \( \alpha \) represents the sampling density and \( m \) is the number of dimensions. A fundamental property of all continuous random processes is that their power spectrum must “roll off” with increasing frequency. The reason for this is that the power of the \( m \)-dimensional underlying continuous process is given by

\[
\sigma_R^2 = \frac{1}{(2\pi)^m} \int_{-\infty}^{\infty} d\omega_1 \cdots \int_{-\infty}^{\infty} d\omega_m \cdot \Gamma_R(\omega)
\]

If the PSD does not decay sufficiently quickly to zero, this integral will indicate an infinite power; of course, no physical process can have infinite power.

In natural images, the PSD actually decays very quickly to zero. Empirical evidence (as well as some interesting thought experiments) show that the PSD of a large class of images typically decays as

\[
\Gamma_X(\omega_1, \omega_2) \propto \frac{1}{\|\omega\|^p} = \frac{1}{(\omega_1^2 + \omega_2^2)^{p/2}},
\]

where \( p \) is usually in the range 2 to 3. Essentially, this means that images are strongly “low pass” in nature. Take a look around you and notice that there are usually many large smooth regions in a scene, with intervening edges. Even an ideal step edge has a PSD which rolls off as \( \frac{1}{\omega} \) in the direction perpendicular to the edge. Stronger high frequency components are created by texture, but if you look closely enough at a textured surface you will find that at some scale the components of the texture become smooth.
5.2 The Periodogram as a PSD Estimator

Let \( X[n] \) be a one dimensional random process, and let \( x[n] \) denote a realization (or observation) of this process. We have access only to a limited number of observations. In particular, suppose we have access to the \( N \) observations \( x[0] \) through \( x[N-1] \). Based on these observations, we wish to compute an estimate for the PSD of the underlying random process. One common approach is to compute the \( N \)-point DFT (or FFT) of the available observations and square their magnitude. The purpose of this sub-section is to explain how this method, known as the periodogram, relates to the true power spectrum \( \Gamma_X(\omega) \).

To begin, let us define a sequence \( y[n] \) of infinite extent, formed by

\[
y[n] = \begin{cases} 
  x[n] & 0 \leq n < N \\
  0 & \text{otherwise}
\end{cases}
\]

In other words, \( y[n] \) contains our set of \( N \) observations, with 0’s at all other locations. Equivalently, we could view \( y[n] \) as the result of applying a window function to the entire realization \( x[n] \). That is,

\[
y[n] = \Pi[n] \cdot x[n],
\]

where

\[
\Pi[n] = \begin{cases} 
  1 & 0 \leq n < N \\
  0 & \text{otherwise}
\end{cases}
\]

Now consider the following auto-correlation estimate:

\[
R_{\text{biased}}^{x}[p] = \frac{1}{N} \sum_{n} y[n] y[n - p]
\]

(23)

Note that at lag \( p \), this estimator sums the products \( x[n] x[n - p] \) over \( N - |p| \) values for \( n \), and then divides by \( N \). We call this a biased estimator, since a true average would divide by \( N - |p| \) instead of \( N \). We are using time averages to approximate the statistical averaging of the expectation operator in \( R_{X}[p] = E[X[n]X[n-p]] \), but our averages are also biased by the factor

\[
\Lambda[p] = \begin{cases} 
  1 - \frac{|p|}{N} & -N < p < N \\
  0 & \text{otherwise}
\end{cases}
\]

It is worth noting that this biasing function is just the convolution of the window function \( \Pi \) by itself, divided by \( N \).

If \( N \) is very large, we can expect that \( R_{\text{biased}}^{x}[p] \) will be a reasonable estimate for \( R_{X}[p] \), at lags \( p \) which are much smaller than \( N \). We could consider removing the bias by replacing \( \frac{1}{N} \) with \( \frac{1}{N-|p|} \) in equation (23). However, this would amplify the most unreliable auto-correlation estimates – the ones with large lags, which are formed by averaging \( x[n] x[n - p] \) over only a few different values for \( n \). This, in turn, would create substantial noise in the power spectrum estimate formed by taking the Fourier transform of the estimated auto-correlation sequence. Essentially, then, \( R_{\text{biased}}^{x}[p] \) may be understood as using the triangular window function \( \Lambda[p] \) to suppress the noisy estimates at high lags.
As foreshadowed in the foregoing paragraph, we can form an estimate for the PSD by taking the Fourier transform of $R_{x}^{biased} [p]$. This estimate is

$$\Gamma_{x}^{biased} (\omega) = \sum_{p} e^{-j\omega p} \frac{1}{N} \sum_{n} y[n] y[n-p]$$

$$= \frac{1}{N} \sum_{p} e^{-j\omega p} (y * \tilde{y}) [p]$$

$$= \frac{1}{N} \tilde{y}(\omega) \tilde{y}^* (\omega)$$

$$= \frac{1}{N} |\tilde{y}(\omega)|^2$$

Here, we have used the fact that the Fourier transform of $\tilde{y}[n] = y[-n]$ is the complex conjugate of $\tilde{y}(\omega)$. To complete this introduction, we define the periodogram of our $N$ observations to be normalized squared magnitudes of the $N$-point DFT coefficients, i.e.,

$$P_{x} [k] = \frac{1}{N} |Y[k]|^2 = \left[ \frac{1}{N} |\tilde{y}(\omega)|^2 \right]_{\omega=(\frac{2\pi}{N}k) \mod 2\pi} = \Gamma_{x}^{biased} (\omega)_{\omega=(\frac{2\pi}{N}k) \mod 2\pi}$$

In conclusion, the periodogram is a frequency-sampled version of the biased power spectrum estimate formed by taking the Fourier transform of the biased auto-correlation estimate $R_{x}^{biased}$.

For multi-dimensional signals, everything is essentially the same. If $X[n]$ is an $m$-dimensional WSS random process and we have $N^m$ observations $x[n]$, at locations $n \in [0, N)^m$. The periodogram estimate of the PSD is formed by squaring the magnitudes of the DFT coefficients, yielding

$$P_{x} [k] = \frac{1}{N^m} |Y[k]|^2 = \Gamma_{x}^{biased} (\omega)_{\omega=(\frac{2\pi}{N^m}k) \mod 2\pi}$$

### 5.3 Problems with the Periodogram

We have already seen that the periodogram is a biased estimate of the true power spectrum. In the time domain, this bias arises from a multiplication of the auto-correlation estimates by the triangular window function, $\Lambda[p]$. In the frequency domain, this means that the power spectrum has been convolved by

$$\hat{\Lambda}(\omega) = \left( \frac{\sin \omega N/2}{\sin \omega/2} \right)^2$$

This convolution limits the inherent resolution of the estimated power spectrum, in the frequency domain. Another way to see that there is limited spectral resolution is to observe that all of the information in $\Gamma_{x}^{biased} (\omega)$ is captured by its samples, at the locations $\omega = \frac{2\pi}{N}k \mod 2\pi$.

Unfortunately, the periodogram has a much more serious problem. Even though the biased auto-correlation estimate behind this spectral estimator has
lower variance (less noise) than an unbiased auto-correlation estimate, the variance of $\Gamma_{x}^{\text{biased}} (\omega)$ is still very large. In fact, the variance of $\Gamma_{x}^{\text{biased}} (\omega)$ does not decrease, as the number of available samples $N$ increases!! For Gaussian random processes, it can be shown that

$$\text{var} [\Gamma_{X}^{\text{biased}} (\omega)] = \Gamma_{X}^{2} (\omega) \left[ 1 + \left( \frac{\sin \omega N}{N \sin \omega} \right)^{2} \right] \quad \longrightarrow N \to \infty \Gamma_{X}^{2} (\omega)$$

Thus, the root-mean-square deviation in the power spectral estimate is as large as the estimate itself!

A reasonable conclusion from the above is that taking the DFT of a finite collection of samples does not produce a reliable estimate of the underlying power spectrum at all. Moreover, the reliability is not improved by increasing the number of samples (i.e., the number of points in the DFT). This is rather counter-intuitive, so it is easy to fall into the trap of using the DFT to estimate power spectra and assuming (wrongly) that having a long enough sequence will fix up any problems.

5.4 A Simple Fix to the Periodogram Approach

The real source of the periodogram problem described above is that we are trying to use all of the available samples to get as much spectral resolution as possible. If the number of available samples $N$ is doubled, the periodogram effectively applies a $2N$-point DFT, which samples the spectrum twice as finely, but each spectral sample is no more reliable than before. Another way to see this same thing is that as $N$ doubles, we double the width of the triangular window function $\Lambda [p]$, hence halving the width of $\hat{\Lambda} (\omega)$ which is “blurring” the spectral estimate.

In order to overcome the total lack of reliability of the periodogram’s spectral estimates, we will have to sacrifice resolution in order to reduce the variance of the estimates. Many strategies have been proposed for doing this, the simplest of which is the Bartlett method, described below.

The Bartlett method is perhaps the most obvious approach to trading spectral resolution for accuracy. Rather than devoting all $N$ samples to improving the spectral resolution, we divide the $N$ samples into disjoint segments of length $L$. Suppose there are $Q$ such segments, so that $N = LQ$. Let $x_q [n]$ denote the sequence of $L$ samples belonging to the $q^{\text{th}}$ segment and let $\Gamma_{x,q} (\omega)$ denote its periodogram. The spectral estimate is then taken to be the average of all $Q$ periodograms. That is,

$$\Gamma_{x}^{\text{bartlett}} (\omega) = \frac{1}{Q} \sum_{q=0}^{Q-1} \Gamma_{x,q} (\omega)$$
In the sampled frequency domain, this means that
\[
\Gamma_{x}^{\text{bartlett}}(\omega)\big|_{\omega = \frac{2\pi k}{N} \mod 2\pi} = \frac{1}{Q} \sum_{q=0}^{Q-1} \frac{1}{L} |X_q[k]|^2, \quad 0 \leq k < L
\]
\[
= \frac{1}{N} \sum_{q=0}^{Q-1} |X_q[k]|^2, \quad 0 \leq k < L
\]

Not surprisingly, the variance of the Bartlett spectral estimates is proportional to \(\frac{1}{Q} = \frac{L}{N}\). Thus, for example, if we fix the value of \(L\), using \(L\)-point DFT’s, no matter how large the actual number of samples \(N\), the variance of the spectral estimates will decrease in proportion to \(N\), while the bias (the width of the \(\hat{\Lambda}(\omega)\) blurring term) will be independent of \(N\).

In multiple dimensions, the Bartlett method takes the available observations in disjoint blocks, or volumes, calculating the periodogram of each such block and averaging the results.

References