AUTO-TUNE

From 1976 through 1989, Dr. Andy Hildebrand worked for the oil industry, interpreting seismic data. By sending sound waves into the ground, he could detect the reflections and map potential drill sites. Dr. Hildebrand studied music composition at Rice University and then developed audio processing tools based on his knowledge in seismic data analysis. He was a leading developer of a variety of plug-ins, including MDT (Multiband Dynamics Tool), JVP (Jupiter Voice Processor), and SST (Spectral Shaping Tool). At a dinner party, a guest challenged him to invent a tool that would help her sing in tune. Based on the phase vocoder, Hildebrand's Antares Audio Technologies released Auto-Tune in late 1996.

Auto-Tune was intended to correct or disguise off-key vocals. It moves the pitch of a note to the nearest true semitone (the nearest musical interval in traditional, equal temperament Western tonal music), thus allowing the vocal parts to be tuned. The original Auto-Tune had a speed parameter which could be set between 0 and 400 milliseconds, and determined how quickly the note moved to the target pitch. Engineers soon realized that by setting this "attack time" very short, Auto-Tune could be used as an effect to distort vocals, and make it sound as if the voice leaps from note to note in discrete steps. It gives the voice an artificial, synthesiser like sound that can be appealing or irritating depending on taste. This unusual effect was the trademark sound of Cher's 1998 hit song, "Believe."

As with many audio effects, engineers and performers found a creative use, quite different from the intended use. As Hildebrand said, "I never figured anyone in their right mind would want to do that" [57]. Yet Auto-Tune and competing pitch correction technologies are now widely applied (in amateur and professional recordings, and across many genres) for both intended and unusual, artistic uses.

frame, many new effects are possible that are not easily implemented in the time domain.

Figure 8.1 shows a diagram of the phase vocoder process. The following steps are common to every phase vocoder effect. Each step is discussed in detail in a subsequent section.

1. Given an input signal of arbitrary length, choose a *frame* of N consecutive samples. The value N is known as the *frame size* or *window size*.
2. Multiply the signal by a *window function* of length N. The window function is defined to have a nonzero value for N consecutive samples and a value of zero everywhere else. By multiplying the input

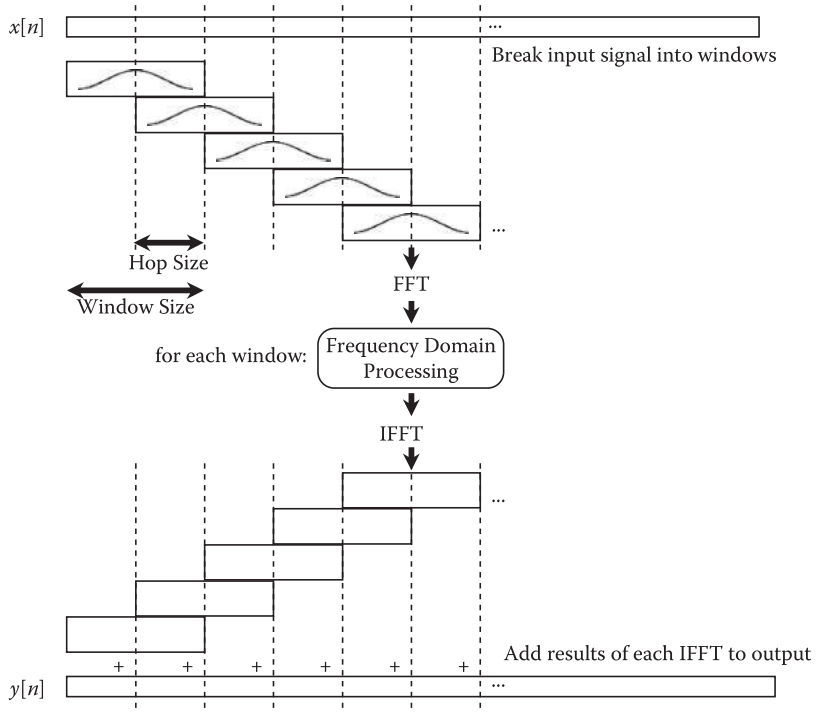

FIGURE 8.1
Overview of the overlap-add process for phase vocoder effects.

signal and the window function, only the N samples in the frame will remain; the rest will be set to zero.

3. After the signal has been *windowed*, apply a *fast Fourier transform* (FFT). Since the windowed signal contains N points, an FFT of size $M \geq N$ must be used. Typically the window size and FFT size are identical; however, in some cases shorter windows can be used with larger FFTs. In practice, the FFT size M is almost always a power of 2 for reasons of computational efficiency. The combination of window function and FFT constitutes the short-time Fourier transform.
4. The output of the FFT will be a collection of M frequency domain *bins* containing *magnitude* and *phase* information for each frequency. Each phase vocoder effect will apply a different type of processing in this step, as discussed below in the "Phase Vocoder Effects" section.
5. Apply the *inverse fast Fourier transform* (IFFT) to the output of step 4. This will once again produce M samples in the time domain.
6. Add the M samples from step 5 to the *output buffer*, which holds the output signal from the effect.

7. Move on to the next frame: move forward H samples in the input signal and return to step 1. The output in step 6 will also move forward H samples. The value H is called the *hop size*. The hop size is sometimes equal to the window size but is frequently a fraction of it (e.g., $H = N/2$ or $H = N/4$).

This process is known as *overlap-add*; it works by analyzing a set of overlapping frames within the input signal, and the output of each frame after processing, properly aligned in time, is added to form the output signal. The following sections discuss each aspect of the overlap-add process.

Windowing

It is important to note that the phase vocoder does not analyze the frequency content of the entire signal at once. Instead, the STFT analyzes only the frequency content present in a particular frame located at a specific time within the signal. A window function of length N will contain N nonzero samples starting from $n = 0$, with a value of 0 at all other samples. The simplest type is the *rectangular* window:

$$w_{rect}[n] = \begin{matrix} 1 & 0 \le n < N \\ 0 & |n| \ge N \end{matrix} \tag{8.1}$$

Many types of windows are possible. Other common variants are the *Bartlett* (triangular) window, the *Hann* window, and the *Hamming* window (Figure 8.2):

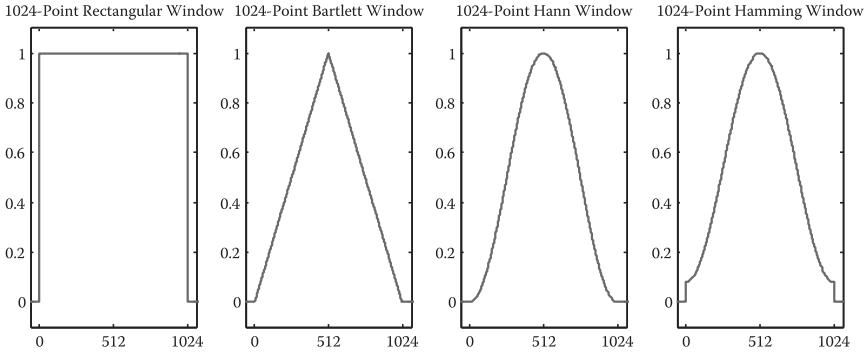

FIGURE 8.2
Four popular window functions.

$$
\begin{aligned}
w_{bart}[n] &= \begin{cases} 1 - \left| \dfrac{2n}{N-1} - 1 \right| & 0 \le n < N \\ 0 & |n| \ge N \end{cases} \\
w_{hann}[n] &= \begin{cases} 1 - \cos\dfrac{2\pi n}{N-1} \Big/ 2 & 0 \le n < N \\ 0 & |n| \ge N \end{cases} \\
w_{hamm}[n] &= \begin{cases} 0.54 - 0.46\cos\dfrac{2\pi n}{N-1} & 0 \le n < N \\ 0 & |n| \ge N \end{cases}
\end{aligned}
\tag{8.2}
$$

Intuitively, the motivation for choosing different types of windows relates to smoothing the edges of the windowed signal. As Figure 8.3 shows, the rectangular window cuts off abruptly at the start and end, and these discontinuities will create undesirable effects known as *sidelobes* in the frequency domain, where energy at one frequency will bleed into other frequency bins. The triangular, Hann, and Hamming windows produce more gradual transitions at the edges, which reduce the sidelobe amplitude. However, they do so at the cost of reducing the precision with which any given frequency component can be identified. In signal processing, the trade-off is described as *main lobe width* versus *sidelobe height*; a more complete discussion can be

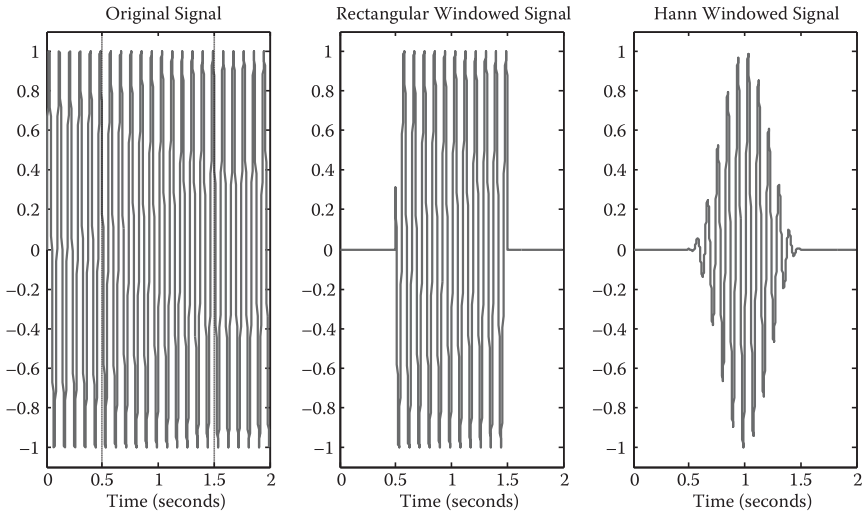

FIGURE 8.3
The effect of a rectangular window and a Hann window applied to a signal.

found in [2]. Though there is no single best solution for all cases; in phase vocoder audio effects Hann and Hamming windows are frequently used.

To apply a window function, the input signal is multiplied by the window. The window function can be offset in time to isolate different regions of the input signal:

$$x_{window}[m,n] = x[n]w[n-m] \tag{8.3}$$

where m is the offset of the window. The samples from m to $m + N - 1$ will be isolated by the window function, with all other samples set to zero.

Analysis: Fast Fourier Transform

The FFT is an algorithm for calculating the *discrete Fourier transform* (DFT) of a signal. In turn, the DFT takes M time domain points as input and produces M regularly spaced frequency domain bins as output. Converting from the time domain to the frequency domain in this way is known as the *analysis* step of the phase vocoder (in contrast to the *synthesis* step going from frequency to time domain). It is important that the size M of the FFT be at least as large as the length of the input signal. Since real-world audio signals can be of arbitrary length, this illustrates the importance of the window function in isolating only a small number of points for the FFT.

We can represent the combination of window and FFT (together, the STFT) as follows:

$$X[m,k] = \sum_{n=-\infty}^{\infty} x[n]h[m-n]e^{-j2\pi nk/M} \tag{8.4}$$

where n and k represent the short-time frame start and frequency bins, respectively. Here k ranges from 0 to $M - 1$, representing evenly spaced frequency bins between 0 and 2π.

Interpreting Frequency Domain Data

Audio signals are always real numbers, but the values of the frequency bins are complex, containing a real and an imaginary component, $x + jy$. Both components are crucial to making sense of the frequency data, but their meaning is easier to understand in *polar* (or *magnitude phase*) representation, $Ae^{j\phi}$. We can convert the polar representation and the *Cartesian* real-imaginary form,

$$\begin{aligned} &x + jy = Ae^{j} \;\rightarrow \\ &A = |x + jy| = \sqrt{x^2 + y^2}\,; \phi = \arg(x + jy) = \operatorname{atan2}(y, x) \end{aligned} \tag{8.5}$$

where atan2 is a version of the arctangent function, available in most programming languages, which produces values in the range $(-\pi, \pi]$ depending on the quadrant of x and y (see Chapter 1). So, the time frequency bins in Equation (8.4) can be written as

$$X[m,k] = |X[m,k]|e^{j\phi[m,k]} \tag{8.6}$$

One important use of phase information is to improve the resolution of frequency detection. Recall that the FFT produces only a fixed number of frequency bins, but signals may contain frequencies that fall between the bin frequencies. The following section shows how comparing phase from two consecutive frames can be used to recover exact frequencies between the bins.

Target Phase, Phase Deviation, and Instantaneous Frequency

For frequency bin k, a target phase, ϕ_t, can be calculated using the bin frequency and the unwrapped bin phase of the previous hop. This target phase is the sum of the previous unwrapped phase and the expected phase increment. The expected increment is the frequency of the sinusoid multiplied by the hop size. So we have

$$\phi_t[n,k] = \phi[n-1,k] + \omega_k h \tag{8.7}$$

where ω_k is the frequency of bin k and h is the hop size. This target phase represents the perfect case of a steady-state sinusoid fitting exactly into a frequency bin, i.e., no spectral leakage. Since this is almost never the case, we have some phase deviation ϕ_d, as shown in Figure 8.4 and given by

$$\phi_d[n,k] = \phi[n,k] - \phi_t[n,k] \tag{8.8}$$

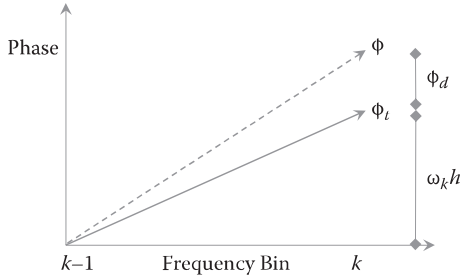

FIGURE 8.4
Relationship between actual phase and target phase. The instantaneous frequency is represented by the gradient of the dashed line, and the bin frequency is the gradient of the solid line.

The corresponding value in the range (–π:π], known as the principal argument, of this deviation phase is then used to calculate the unwrapped phase increment per hop, $\Delta\phi_h$:

$$\phi_h[n,k] = \omega_k h + \arg\phi_d[n,k] \tag{8.9}$$

The instantaneous frequency, f_i, can then be calculated:

$$f_i[n,k] = \frac{\phi_h[n,k]}{2\pi h} \cdot f_s \tag{8.10}$$

where f_s is the sample frequency. This instantaneous frequency can then be viewed as a more accurate frequency measurement than the frequency resolution offered by the filterbank or STFT.

Synthesis: Inverse Fast Fourier Transform

The specific processing done in the frequency domain depends on the particular phase vocoder effect. Once this processing is complete, however, all phase vocoder effects will convert the signal back to the time domain using the inverse fast Fourier transform (IFFT):

$$x[m] = \frac{1}{M}\sum_{k=0}^{M-1} |X[m,k]| e^{j(2\pi km/M + \phi[m,k])} \tag{8.11}$$

This process is known as *synthesis* or *reconstruction*. It produces M points in the time domain, which can be added to the output buffer. Depending on the type of frequency domain processing performed, it is possible that the output frame may not have smooth edges even if a Hamming window (or similar) was used for analysis. Occasionally then, applying a second window before adding the result to the output buffer will help reduce audible artifacts.

Overlap-Add

The purpose of the phase vocoder process is to analyze the frequency content of each short frame, modify it, and reconstruct it in the time domain. The manner in which the frames advance from one to the next is important to the proper operation of the phase vocoder. Once a frame has been analyzed, processed, and synthesized, the effect should advance by the *hop size*. Suppose the hop size is given by H. Then if the window in frame i previously began at sample m, the window in frame $i + 1$ will begin at sample $m + H$. The hop

size is always less than or equal to the window size; if it were greater than the window size, there would be a gap between successive windows. Smaller hop sizes (more overlap) will often produce better-quality output in many phase vocoder effects; however, they require more computation.

Hop sizes of one-half, one-fourth, or one-eighth the window size are common. A useful guideline for choosing hop size is the *constant overlap-add* (COLA) *criterion* [2]. Suppose that in the steps listed above for the phase vocoder theory overview, no processing at all is done in the frequency domain (step 4), and instead, the FFT is immediately followed by the inverse FFT. For windows and hop sizes meeting the COLA criterion, the output signal at the end of the overlap-add process would be identical to the input signal, with no distortion or modulation introduced by the windowing process. Essentially, this requires that the window functions, when overlapped, add to a constant value, as depicted in Figure 8.5. Rectangular windows meet the COLA criterion for a hop size equal to the window size; Bartlett, Hann, and Hamming windows require a hop size of at most half the window size. Integer divisions of this maximum hop size are also possible: for example, a hop size of one-eight the window size will meet the COLA criterion for all four windows.

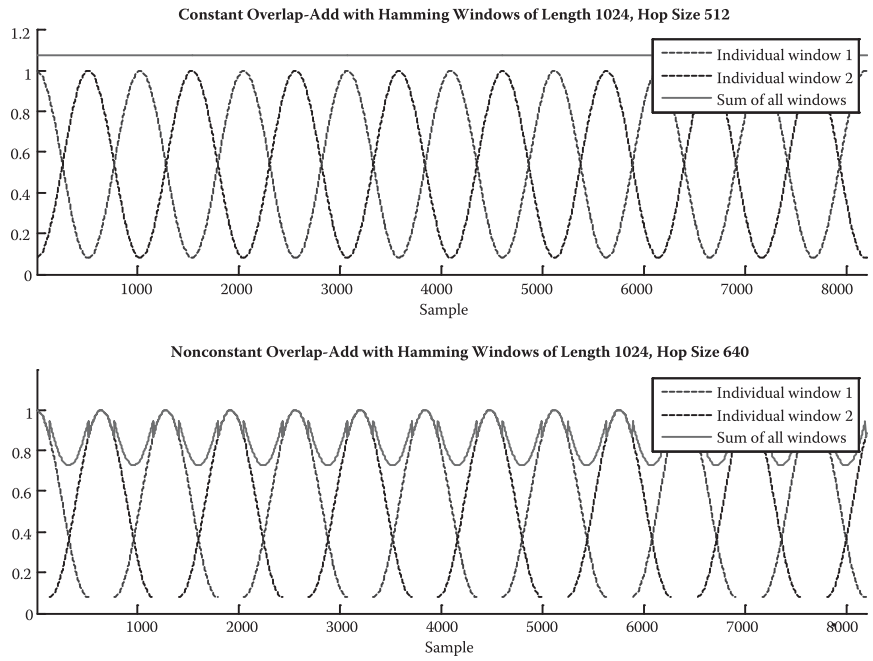

FIGURE 8.5
The constant overlap-add criterion requires that the window functions, when overlapped, add to a constant value. It holds in the top plot, but not the bottom plot.

To summarize, it is important to choose a window and hop size such that the windowing process itself does not introduce unnecessary artifacts into the output. The exact choice of window type, length, and hop size depends in large part on the specific effect, several of which are discussed below.

Filterbank Analysis Variant

Implementation of the phase vocoder is also possible using a filterbank approach. This leads to a computationally more expensive implementation, but it can be shown to be theoretically equivalent, and it is often straightforward to consider the phase vocoder as a filterbank.

Returning to the STFT equation, Equation (8.4), it is clear that

$$X[n,\omega] = h[n] * x[n]e^{-j\omega n} \tag{8.12}$$

where ω is now used to represent $2\pi k/N$.

This can be seen as a demodulation of the signal components at frequency ω down to baseband, followed by a low-pass filtering of the signal using the filter $h[n]$. This is known as the complex baseband filterbank implementation, and is illustrated in Figure 8.6, top.

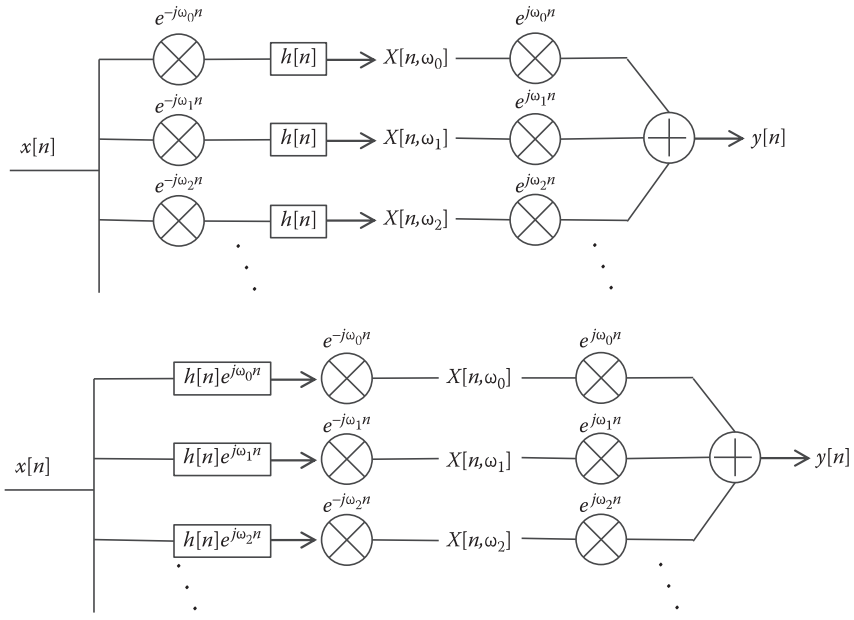

FIGURE 8.6
The complex baseband filterbank implementation (top) where $h[n]$ is a low-pass filter, and the complex band-pass implementation (bottom). The filters for the first three bins only are shown.

If the variables are switched in Equation (8.4) such that $m \rightarrow n - m$, we obtain

$$X[n, \omega] = e^{-j\omega n} \sum_m x[n-m]h[m]e^{j\omega m} = e^{-j\omega n} \quad x[m] * \left(h[n]e^{j\omega m}\right) \tag{8.13}$$

Now we can view the filter as being band-pass at the frequency ω. After filtering, the result is then demodulated back down to baseband. This implementation is known as the complex band-pass implementation. It is illustrated in Figure 8.6, bottom.

For the signal to be reconstructed from the magnitude and phase values of $X(n, w)$, each baseband signal must be modulated back to the frequency ω.

$$y_k[n] = e^{jw_k n} X\left(n, w_k\right) = \left|X\left(n, w_k\right)\right| e^{j \ w_k n + \phi(n, w_k)} \tag{8.14}$$

The signal $y[n]$ is reconstructed by summing these terms for each frequency bin.

Oscillator Bank Reconstruction Variant

Since we can assume that the time domain signal $x[n]$ is real, frequency bins that are symmetric about the Nyquist frequency will be conjugate pairs.

$$X[n, k] = X^*[n, N-k] \tag{8.15}$$

These two signals may then be summed to simplify the analysis. This also results in a more meaningful interpretation:

$$\begin{aligned} \hat{y}_k\left(n\right) &= \left|X\left(n, k\right)\right| \quad e^{j\phi(n,k)} + e^{-j\phi(n,k)} \quad e^{jw_k n} + e^{-jw_k n} \\ &= 2\left|X\left(n, k\right)\right| \cos\left(w_k n + \phi_k\left(n, w_k\right)\right) \end{aligned} \tag{8.16}$$

Oscillator bank reconstruction is clearly computationally expensive. Despite this, it is generally a musically intuitive way of viewing the synthesis stage, especially for musicians. Using an IFFT overlap and add approach is much more efficient. The STFT represents a downsampled version of the filterbank outputs. This is possible due to the filtering step band-limiting the channel signals.

Phase Vocoder Effects

The phase vocoder and its underlying theory have a wide range of applications. In the fields of machine listening and music informatics, these include