Real-Time Speech Pitch Shifting on an FPGA

Frequency-Domain Pitch Shifting

Like the time-domain technique, the frequency-domain technique is based on shifting small overlapping windowed blocks of data in time and resampling. To shift down in pitch, the overlapping data blocks are shifted closer together to create a signal with the same pitch but shorter duration. The signal is then resampled, expanding to the original duration and stretching the signal in time to achieve a decrease in pitch. To raise the pitch, data blocks are spread further apart in time to create a signal longer in duration. Again, the signal is resampled to restore the original duration, which compresses the signal in time to achieve an increase in pitch.

The frequency-domain technique continues with the assumption that speech is short-time stationary, that is, periodic with relatively constant frequency components over small ranges of time. As such, adjacent blocks will effectively have the same frequency content. However, when these blocks are shifted in time, the block transitions will not be in phase. The resulting phase discontinuities introduce noise, just as we saw in the purely time-domain technique. However, by utilizing frequency domain information, the phase discontinuities can be eliminated.

The phase vocoder algorithm offers an effective solution to the problem of phase discontinuity [4]. It is considered a frequency-domain technique because it utilizes the Short-Time Fourier Transform (STFT), a common audio processing tool that involves taking the Discrete Fourier Transform (DFT) of short, periodic blocks of an audio signal. By properly modifying the phase terms of the STFT and re-synthesizing the time-domain data, it is possible to match the phases of each frequency component across block transitions. Figure 6 demonstrates the phase discontinuity problem introduced by shifting overlapping blocks.

In addition to modeling audio as stationary and periodic over small blocks, the phase vocoder makes several other assumptions. Namely, sound can be modeled as a sum of sinusoids, and each frequency bin in the STFT of an audio block contains no more than one sinusoidal component. In other words, the algorithm operates as if any non-zero frequency bin magnitude is caused a single tone corresponding to the range of frequencies spanning that bin. These assumptions allow us to represent each STFT frequency bin as a single sinusoidal component of the signal. We can then modify the phase offset associated with the sinusoid to ensure continuity when added to the same component of the previous block.

We can summarize the phase vocoder algorithm with the following steps:

1. Calculate the STFT of overlapping signal blocks
2. Modify the phase of each STFT bin
3. Re-synthesize (IFFT) the block and apply the appropriate time shift (see Figure 7)
4. Resample block to restore original duration

To determine how to modify the STFT phase terms, we must derive a phase propagation formula that will accurately predict how the phases of each frequency bin will progress in time. There are several factors that will effect phase propagation: (1) block size; (2) amount of time shift required following re-synthesis; and (3) actual frequency of bin sinusoid (not necessarily the center bin frequency).

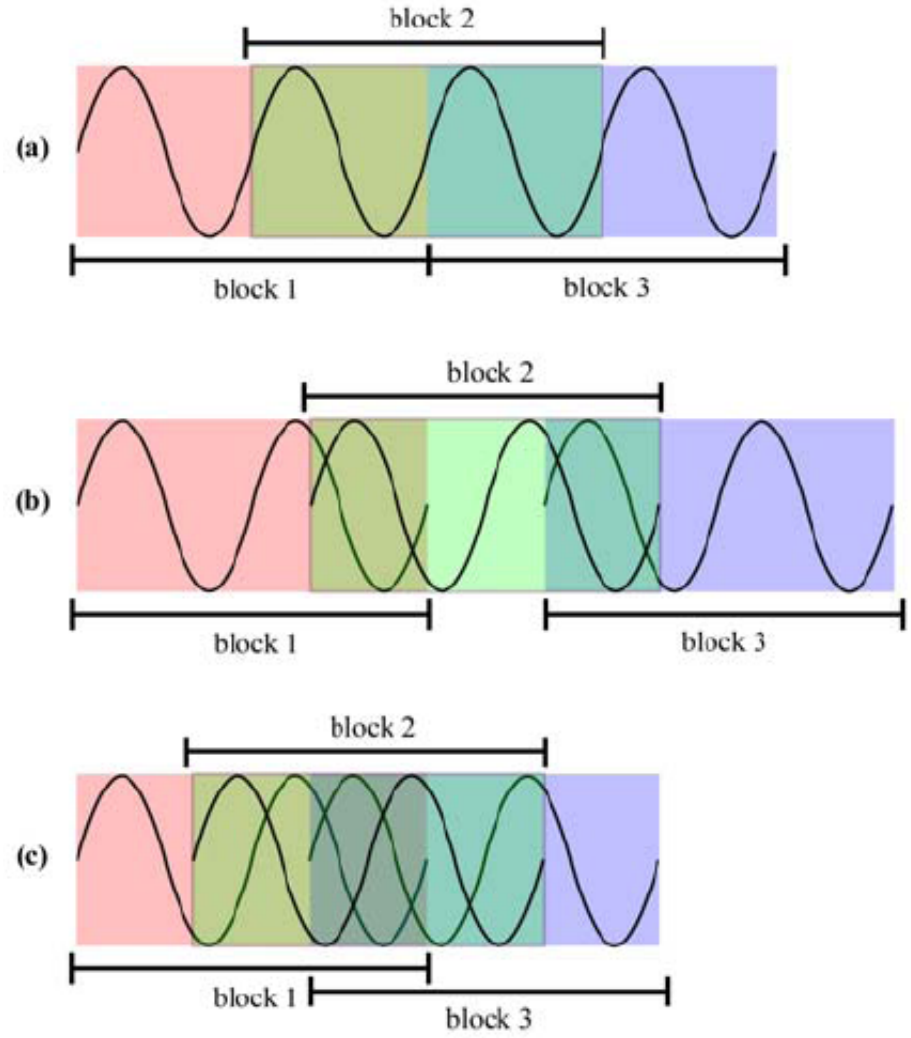

Figure 6. Time shifting of overlapping blocks; (a) depicts an input signal split into 3 overlapping blocks; (b) blocks are shifted forward in time to increase signal duration; (c) blocks are shifted back in time to decrease signal duration

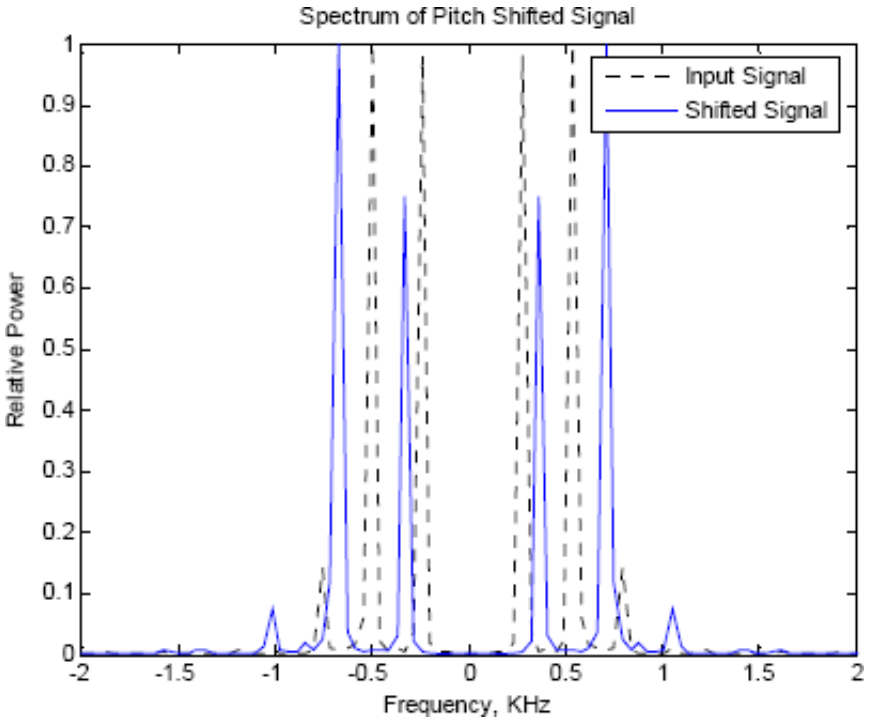

Figure 7. Input and shifted frequency spectrum using phase vocoder. Note slight noise introduced around 1.5KHz and higher.

Many implementations of the phase vocoder have been developed in audio processing literature. We will use the symbols and phase propagation formula proposed by Loroche and Dolson [5].

The process of calculating STFT's of signal blocks will be referred to as the analysis step, while the inverse will be called the synthesis step. During the analysis step, the u^{th} block will start at time $t_a^u = uR_a$, where R_a will be called the analysis hop factor and represent the size of the block. Using lower-case for time-domain signals and upper-case for frequency-domain representations, $x(n)$ and $y(n)$ will be the discrete-time input and output signals respectively. Both analysis and synthesis steps will utilize windows, $h(n)$ and $w(n)$ respectively. We can now define the STFT analysis step and the time domain output following the synthesis step:

$$X(t_a^u, \Omega_k) = \sum_{n=-\infty}^{\infty} h(n)x(t_a^u + n)e^{-j\Omega_k n} \quad \text{(Eq. 3)}$$

$$y(n) = \sum_{u=-\infty}^{\infty} w(n - t_s^u)y_u(n - t_s^u) \quad \text{(Eq. 4)}$$

where,

$$y_u(n) = \frac{1}{N}\sum_{k=0}^{N-1} Y(t_s^u, \Omega_k)e^{j\Omega_k n} \quad \text{(Eq. 5)}$$

Time scaling is effected by shifting the output blocks, such that $t_s^u = uR_s$ when $R_a \neq R_s$. We are left only to define the phase propagation equation by which the phase of the output STFT, $Y(t_s^u, \Omega_k)$, will be altered. Omitting the full derivation, Laroche and Dolson [5] yield the following three equations, (Eq. 8) representing phase propagation:

$$\Delta\Phi_k^u = \angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) - R_a\Omega_k \quad \text{(Eq. 6)}$$

$$\hat{\omega}_k(t_a^u) = \Omega_k + \frac{1}{R_a}\Delta_p\Phi_k^u \quad \text{(Eq. 7)}$$

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^{u-1}, \Omega_k) + R_s\hat{\omega}_k(t_a^u). \quad \text{(Eq. 8)}$$

Result (Eq. 6) estimates phase propagation based on the center bin frequency (i.e. $\Omega_k = k/N$). The actual frequency of the sinusoid contained in the k^{th} bin is estimated by (Eq. 7) based on the difference in bin phase propagation rate between blocks. We have implemented a phase vocoder according to these results. MATLAB code is included as Appendix D. The final step, resampling, is performed in the same way as the time-domain technique.

Simulations demonstrate that most of the noise produced by the time-domain pitch shifting technique is eliminating by altering the phases of the STFT. Figure 7 depicts the input and output signal spectrums. However, the frequency-domain method, as described here, is far from perfect. Notably, our simulation suffers from the typical "distance" and "phasiness" issues commonly associated with the phase vocoder. [5] proposes several methods for correcting these phase errors, but they are out of the scope of this project since the simulation produced results of acceptable fidelity.