Phase Vocoder Algorithm For Pitch Shifting

Warren L. G. Koontz

May 30, 2019

1 Introduction

This short note provides a rough description of the phase vocoder pitch shifting algorithm. The algorithm is described in the MATLAB documentation[1]. However, this note provides more information about the phase modification step.

The algorithm processes a sequence of blocks of audio samples. In simplest terms, each block of audio samples is processed as follows:

- Compute the short time Fourier transform (STFT).
- Adjust the phase component of the transform.
- Compute the inverse short time Fourier transform (ISTFT)
- Re-sample the recovered audio samples.

The following sections describe the forward and inverse STFT and the rationale and process for adjusting the phase component.

2 Short Time Fourier Transform

This section briefly describes the short time Fourier transform (STFT) and its inverse (ISTFT). The input to the STFT is a block of real-valued audio samples. The number of samples in the block is called the *analysis length*. The output of the STFT is a complex-valued Fourier transform whose length (i.e., number of frequency bins) is called the *window length*. The window length is generally larger than the analysis length and in this implementation

[1] `www.mathworks.com/help/audio/examples/pitch-shifting-and-time-dilation-using-a-phase-vocoder-in-matlab.html`

it is the analysis length time an integer power of 2. The input to the ISTFT is a complex-valued Fourier transform and the output is a block of real-valued audio samples. For this to be the case, the input to the ISTFT must be conjugate symmetrical. In the phase vocoder application, the output of the STFT and the input to the ISTFT have the same length (i.e., the window length). However, number of ISTFT output samples, called the *synthesis length*, can differ from the analysis length.

The STFT is implemented as an object that includes an initialization method to set up the necessary parameters and arrays and a step method to process each block of audio samples. The following MATLAB code fragment describes the heart of the step method:

```
sampleBuffer = [sampleBuffer((analysisLength+1):end) sampleBlock];
fftOutput = fft(sampleBuffer.*sqrt(hann(windowLength,'periodic')));
```

The *sample buffer* is an array whose size is equal to the window length. At each invocation of the step method, the new block of samples is shifted into the sample buffer, shoving out the oldest block. The output is then determined by computing the discrete Fourier transform (DFT) of the product of the sample buffer and a *window function*. The DFT is computed using a version of the fast Fourier transform (FFT) algorithm. The window function is the square root of the periodic Hann function, which is given by

$$h(n) = 0.5(1 - \cos\frac{2\pi n}{N}), \;\; 0 \leq n < N \tag{1}$$

where N is the window length + 1.

The ISTFT is also implemented as an object with an initialization method and a step method. The following MATLAB code fragment describes the heart of the ISTFT step method:

```
sampleBuffer += ifft(fftData).*sqrt(hann(windowLength,'periodic'));
sampleBlock = sampleBuffer(1:synthesisLength)*gain;
sampleBuffer = [sampleBuffer(synthesisLength+1:end) zeros(1,synthesisLengt
```

In the first line, the sample buffer, which is initialized to all zeros, is incremented by the product of the inverse DFT of the input (`fftData`) and the same window function used in the STFT. The output samples are then copied from the sample buffer starting at position 1; the number of output samples is equal to the synthesis length. The samples in the sample buffer are multiplied by a gain factor given by

$$G = L_S \sum_{n=0}^{L_W - 1} h(n) \tag{2}$$

where L_S is the synthesis length and L_W is the window length. Finally, sample buffer is shifted to shove out the output samples and shift in an equal number of zeros. This process is known as *overlap-add*.

What happens if the output of the STFT is passed unchanged to the ISTFT? If the synthesis length matches the analysis length *and* the window length is chosen well, the output of the ISTFT should match the input to the STFT. However, the pitch shifting algorithm includes some meddling with the transformed samples before passing them to the ISTFT. This meddling is described in the next section.

3 Adjusting the Pitch

The pitch adjustment algorithm is based on a tempo adjustment algorithm, i.e., and algorithm to change the tempo without changing the pitch. The tempo change is turned into a pitch change by re-sampling.

Let $X(n),\ 0 \leq n < L_W$ be the output of a step of the STFT and let $A(n)$ and $\phi(n)$ be the magnitude and phase of $X(n)$. The input to the corresponding step of the ISTFT will be $Y(n),\ 0 \leq n < L_W$ with the same magnitude $A(n)$ but a modified phase $\psi(n)$.

Let $\phi_{\text{prev}}(n)$ be the value of $\phi(n)$ from the previous invocation of the STFT step and let

$$\Delta\phi(n) = \phi(n) - \phi_{\text{prev}}(n) \tag{3}$$

This phase change occurs over a time period of L_A sample periods, L_A being the analysis length. We want to alter this phase change in order to vary either the tempo or the pitch of the audio. Unfortunately, both $\phi(n)$ and $\phi_{\text{prev}}(n)$ are "wrapped" to lie in the range $-\pi \leq \phi \leq \pi$ so that $0 \leq |\Delta\phi(n)| \leq 2\pi$. The actual phase change is $\Delta\phi(n) + 2\pi k$ where k is the (unknown) number of wraps. We have to somehow "unwrap" $\Delta\phi(n)$ to find the actual phase change and then modify the unwrapped change.

We start by expressing the wrapped phase change as

$$\Delta\phi(n) = \Delta\phi_{\text{un}}(n) - 2\pi k \tag{4}$$

where $\Delta\phi_{\text{un}}(n)$ is the unwrapped phase change that we seek. The frequency normally associated with the n^{th} bin is $2\pi n/L_W$ radians per sample. If this were the actual frequency of the signal component in this bin, then the total phase change would be

$$\Delta\phi_0(n) = 2\pi n L_A / L_W \tag{5}$$

If we subtract (5) from (4), we have

$$\Delta\phi_{\text{err}} = \Delta\phi_{\text{un}}(n) - \Delta\phi_0(n) - 2\pi k \tag{6}$$

If we assume that the difference between the actual bin frequency and the nominal bin frequency $2\pi n/L_W$ is less than or equal to one half the bin width $2\pi/L_W$, then

$$|\Delta\phi_{\text{un}}(n) - \Delta\phi_0(n)| \leq \pi L_A/L_W \tag{7}$$

This implies that we can isolate the difference $\Delta\phi_{\text{un}}(n) - \Delta\phi_0(n)$ by subtracting an appropriate number of multiples of 2π from $\Delta\phi_{\text{err}}$. For a given angle θ, this can be done as follows:

$$\theta_{\text{unwrap}} = \theta - 2\pi\lfloor\theta/2\pi\rceil \tag{8}$$

The unwrapped angle θ_{unwrap} lies between $-\pi$ and π. Applying this to (6) we have

$$\Delta\phi_{\text{un}}(n) - \Delta\phi_0(n) = \Delta\phi_{\text{err}} - 2\pi\lfloor\Delta\phi_{\text{err}}/2\pi\rceil \tag{9}$$

so that the unwrapped phase change we seek is given by

$$\Delta\phi_{\text{un}}(n) = \Delta\phi_{\text{err}} - 2\pi\lfloor\Delta\phi_{\text{err}}/2\pi\rceil + \Delta\phi_0(n) \tag{10}$$

The following MATLAB script, copied almost directly from MATLAB documentation, performs the calculations described so far:

```
uwdata = 2*pi*alen*(0:(wlen-1))'/wlen;
uwphase = phase-prevphase-uwdata;
uwphase = uwphase-round(uwphase/(2*pi))*2*pi;
uwphase = uwphase+unwrapdata;
```

The next step is to modify the phase change by a factor determined by the desired pitch shift. For a pitch shift of n_s semitones, the factor is given by

$$\alpha = 2^{n_s/12} \tag{11}$$

The resulting modified phase is given by

$$\psi(n) = \psi_{\text{prev}}(n) + \alpha\Delta\phi_{\text{un}}(n) \tag{12}$$

and the input to the ISTFT is given by

$$Y(n) = A(n)\angle\psi(n) \tag{13}$$

Because of the modification, the synthesis length for the ISTFT must also be modified by the factor α, i.e.,

$$L_S = \lfloor \alpha L_A \rceil \tag{14}$$

The output of the ISTFT is a tempo-shifted version of the input to the STFT. Re-sampling from L_S samples to L_A samples restores the tempo and shifts the pitch by the desired amount.