Octave, Detune and Pitch Shifter

VCarrara

Definition

A Pitch Shifter changes the audio signal frequency without changing its time, by either increasing or decreasing the frequency. Although the algorithms presented here can be generalized to work in both directions, they will be separated into 3 similar algorithms: in the first one, the frequency shall be increased, which will be named as Pitch Shifter; in the second the frequency will be decreased, named Detune, while the third one will have the exclusive function of doubling the frequency, named Octave. This separation allows each algorithm to be customized for its functionality and, furthermore, the implementation of different algorithms allows two of them to be used simultaneously, such as, for example, the Pitch Shifter and the Octave.

The usual way to perform a frequency shift is to resample the input signal with the same sampling frequency, but by increasing or by reducing the time interval between samples. When the resample interval is increased in relation to the input signal, there will be an increase (Pitch Shifter) of the output frequency, or a frequency reduction (Detune) when the interval is compressed. To maintain the time duration of the original signal, several windows, or intervals, of the input signal samples are superimposed to compose the tone-altered signal. The algorithms of the three types of effects will be presented below.

Pitch Shifter

For the Pitch Shifter a tonal transformation is adopted, such that a continuously variation parameter s indicates the number of semitones to be increased in the input signal ($0 \leq s \leq 12$), so the frequency ratio between the input and output signal, $p = f_s/f_e$, can be computed by

$$p = 2^{s/12}.$$

From the above equation is easy to see that when p is unity ($s = 0$) the output frequency equals to the input, and when p equals 2 ($s = 12$) the frequency is doubled. Figure 1 shows the input signal at the top, and the output signal at the bottom, which is composed of a combination of two distinct intervals of the input signal, modulated by two sine envelopes out phased by 90 degrees. Thus, while one of the input interval increases in amplitude, the other decreases. On the next interval the increasing signal will then decrease, and a new increasing interval will be added to the output signal. The sine envelopes are shown in the figure, with its increasing part at top and decreasing at bottom. Colors identify the input intervals. In the Pitch Shifter each input window is larger than the re-sampled output signal. Since the increasing and decreasing intervals are symmetric, only one window needs to be formulated. At the end of the decreasing interval the pointer of the input audio buffer is stored in an integer variable j_i. At this time the input and output signal are synchronized with no delay. The length of each window is given by m_e, which corresponds to an output window with half size, m_s, computed by:

$$m_s = \frac{m_e}{2p}.$$

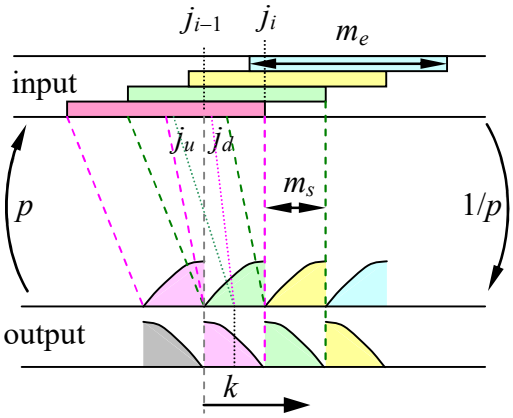

Fig 1 – Time intervals and sample windows of the Pitch Shifter.

In Figure 1 the variable k is a pointer to the output buffer, related to the addresses in both input windows to compute the output signal at this time. k varies between 0 and m_s, with unit increment, at the same sampling frequency of the input signal. When k reaches the half-size window m_s, pointer j_i is incremented by m_s, and k starts the next interval ($k = 0$). Despite the input window size being m_e, j_i is incremented with m_s, which makes the increasing and decreasing windows to overlap, as shown in Figure 1. They overlap each other by ¾ when $s = 12$ and by half size when $s = 0$.

Window lengths in the input signal are converted to window lengths in the output signal by multiplying the length by $1/p$. As p is always equal to or greater than 1, then the time in the output signal is shortened, and the audio frequency is increased. Similarly, an interval length in the output signal corresponds to a length increased by a factor p in the input audio. The dashed lines in the figure indicate this linear transformation, which allows the input audio to be re-sampled as function of k, resulting in pointers j_u and j_d in the input buffer for increasing and decreasing interval, respectively. In fact, from Figure 1 it can be seen that:

$$j_i + m_s - j_u = p\,(2m_s - k)$$
$$j_i - j_d = p\,(m_s - k)$$

which can be rewritten in the form

$$j_u = j_i + m_s\,(1 - 2p) + p\,k$$
$$j_d = j_i - p\,m_s + p\,k$$

Note that the pointers j_u and j_d to the input buffer are always smaller than the output pointer at instant k, given by $j_i + k$. Both j_u and j_d may be or may be not integers. For high sampling frequencies, above 20 kHz, no interpolation in the input buffer shall be required. The maximum delay between input and output signals occurs when the sine amplitudes are at maximum value, which happens at $k = 0$ or $k = m_s$. and is given by

$$\delta t = \frac{m_s}{f_s}(p-1)$$

where f_s is the input sampling rate (normally 44.1 kHz). So the maximum delay depends on the selected tone increment s, and can vary from no delay up to m_s/f_s. Defining now the angle $\varphi(k)$ by

$$\varphi(k) = \frac{\pi}{2}\frac{k}{m_s},$$

the output signal can be computed with

$$u_s = \sin(\varphi(k))\, u_e(j_u) + \sin(\varphi(k) + \pi/2)\, u_e(j_d)$$

in which u_e and u_s are the input and output audio buffers, respectively. Very high window lengths, with m_e above 8000, for a sampling frequency of 44.1 kHz, tend to improve the quality of the output signal, reducing vibrato introduced by the sine envelope. On the other hand, long lengths compromise the synchronicity between input and output, making the delay clearly audible. Short lengths for m_e (less than 4000) tend to introduce undesirable harmonics in the output signal, making it "metallic", but it improves the synchronism. A good compromise between both characteristics can be found with m_e in the range between 5000 and 6000.

Octave

The Octave has the same formulation as the Pitch Shifter, but in Octave the parameter p is constant and equals to 2. The equations of this model are:

$$m_s = \frac{m_e}{4}.$$

and the input buffer pointers will be obtained by

$$j_u = j_i - 3m_s + 2k,$$
$$j_d = j_i - 2m_s + 2k.$$

When $k = 0$, j_u is $3m_s$ samples out of phase with the input signal, while j_d is $2m_s$ samples out of phase. When k is equal to m_s, j_u results m_s samples off phase while j_d is in sync with the input ($j_d = 0$). The sine angle and the composition of the output signal u_s are calculated identically to the Pitch Shifter. A good compromise between time delay and audible artifacts on the Octave output was achieved with 3000 samples for m_e, at 44.1 kHz sampling frequency, which corresponds to a time window of 68 milliseconds, or a time delay of 17 milliseconds.

Detune

The algorithm for Detune effect is similar to the Pitch Shifter, except for the frequency ratio, p, which now is the reciprocal of the Pitch Shifter. Figure 2 shows the input signal at top and the output at bottom. The output signal will again be composed of two intervals, modulated by sine profiles with 90 degrees out of phase. In this algorithm both the increasing input and output window are synchronized at beginning of a new cycle, when j_i is incremented with m_s and k is set to zero. The sampling interval in the input signal has length given by m_e, which corresponds to

$$m_s = p\frac{m_e}{2},$$

samples in the output signal. The variable k will again vary between 0 and m_s, with unit increment, at the same sampling frequency of the input signal. When k reaches the half-sine interval m_s, the j_i pointer is incremented by m_s, and k is reinitialized. Note that, despite the interval in the input signal being m_e, j_i is incremented by m_s, which causes the increasing and decreasing intervals in the input signal to partially overlap, as shown in Figure 2. They overlap by $m_e/2$ samples when $p = 1$ and zero samples when p is 2.

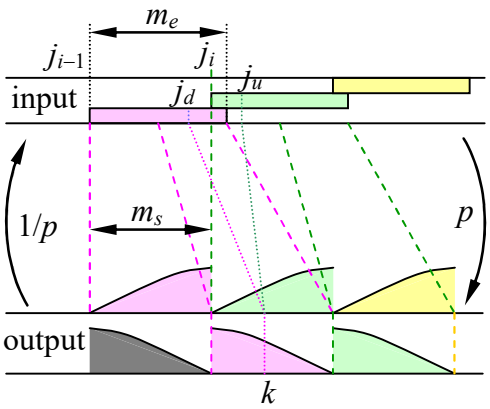

Fig 2 – Time intervals and sample windows of the Detune.

Time intervals in the input signal are converted to time intervals in the output signal by multiplying the interval by p. As p is always equal to or greater than 1, then the time interval in the output is expanded, and the frequency of the input audio is reduced. Analogously, a time interval in the output signal corresponds to a fraction $1/p$ in the input interval. The dashed lines in the figure indicate this linear transformation, which allows obtaining the input samples at position j_u and j_d in the increasing and decreasing intervals, respectively, as a function of the output time k. In fact, from the figure it can be seen that:

$$j_u - j_i = \frac{k}{p}$$

$$j_d - j_{i-1} = \frac{m_s + k}{p}$$

which can be rewritten in the form

$$j_u = j_i + \frac{k}{p}$$

$$j_d = j_i - m_s \frac{p-1}{p} + \frac{k}{p}$$

It can be noted that the pointer to the output buffer is given by $j_i + k$, which will always be greater than or equal to both j_u and j_d. When k is zero, j_u equals to j_i (in sync), while j_d is delayed by $m_s - m_s/p$ samples. When k equals m_s, j_u and j_d are delayed by m_s/p and m_s samples, respectively. The maximum latency can be computed by

$$\delta t = \frac{m_s}{f_s} \frac{p-1}{p}$$

The output signal u_s is computed by the same expression presented in the Pitch Shifter formulation. Best audio quality is achieved with high values of m_e, around 8000 samples. However, latency is noticeable on high tone changing, close to 12, but still acceptable. In this case the time delay (latency) is 91 milliseconds for 44.1 kHz sampling rate.