Audio Transport

Trevor Henderson

May 18, 2017

Abstract

This paper describes a new technique to interpolate between audio signals. This technique uses optimal transport to move the pitches in one signal to the pitches in another. Audibly this sounds like a generalized portamento, or glide between sounds. The main contributions of this paper are methods to remove the different types of phasing distortion that can occur in a naive transport. A real time implementation of the algorithm produces musical results.

1 Introduction

A portamento is musical term used to describe one pitch sliding into another. The portamento can be found in almost all genres of music, whether it be a slur in a vocal passage or in the twang of a slide guitar. It provides smooth way to transition between notes and even chords.

However many instruments Most acoustic keyboard instruments like the piano or organ example can't glide at all. This is fixed with a pitch bend or for monophic instruments with a glide knob. These systems are difficult to move between chords of different shapes. Moreover all of these methods require that the glide happen between sounds of the same timbre.

This paper describes a method for protamento not only between sounds with different timbres but between sounds from different instruments entirely. It does this using optimal transport, a problem mostly found in computer graphics and machine learning applications. It also relies on machinery which is common to phase vocoders.

This paper aims to fix that problem

It functions by

And even instruments that can glide are often limited in the chord shapes they can glide between.

In music the portamento is limi Few instruments for example can glide smoothly between chords of different shapes, For example gliding between chords of different shapes becomes challenging and gliding between chords with different

This paper aims to develop a generalized port

Portamentos can also be polyphonic. For example the slide guitar technique, a technique commonly found in blues and country music, is in essense a polyphonic portamento between chords on the guitar. This paper aims to develop sounds that map between the pitches of audio file.

for example uses this sound The goal of this paper is to

2 Contributions

I developed a new technique to interpolate between two audio signals using optimal transport. The optimal transport problem, discussed in detail later, is defined only for positive real inputs. The spectra of an audio signal however is complex. It is natural that the transport be done on the amplitudes of these complex numbers, but this begs the question of what is to be done with the phase.

This paper describes several types of distortion that can occur because of phasing issues and how to resolve them. These problems are similar to those experienced in phase vocoders — the technology that allows an audio's speed to be changed without changing the pitch. However the solutions posed in this paper improve upon the techniques used in phase vocoders and take advantage of the specifics of this problem.

The new techniques introduced in this paper are a method to segment the spectrum into groups whose local phase relations have audible significance. A method to update the phases of a piece of the spectrum that has been moved to a new pitch. A method to synthesize the audio

3 Related Work

Many audio tools exist that modulate the pitch of an audio signal. The phase vocoder, which became practical in the early 2000's is now pervasive in all types of music production. The phase vocoder is also the technology that enables auto-tune — an effect which swept through the music industry in the mid-2000's and is still popular today.

Several audio tools work to manipulate the pitches within an audio signal seperately. Melodyne is a popular tool in the music industry for correcting the pitches of notes. In 2008, Melodyne released a feature to adjust the pitches of individual notes in polyphonic music, allowing a user to rewrite the harmony of a song. This technology works on a static audio file, but in 2012 a product named QQQQ allowed for individual pitch manipulation to happen in real time.

However neither of these products pitch shift on with granularity that this method does. In addition they rely on the user to select notes and decide where they want to go. While this is an extremely useful feature in the studio, it is hard to control in the live setting. This method allows the user to simply specify where they would like the pitches to go with another audio file and they can go there. http://www.zynaptiq.com/pitchmap/

Optimal transport has been used for audio processing before. However so far as I can tell, no paper has synthesized sound with optimal transport. Most of the existing papers in this cross section use optimal transport to help transcribe music. By using a distance metric based on the harmonic series seems to be useful for determining where a note exists.

4 Optimal Transport

Discussion of optimal transport.

Distance.

It also allows for you to interpolate between the two shapes along the trajectory.

In 1 dimension it is a solved problem.

For any dimensionality higher than 1, fast computation of the wasserstein2 distance is

Optimal transport is however only defined for, nonnegative masses. Audio has phase. Initial attempts to tackle the problem of phase involved formulating the transport problem differently. But you can't hear the distance between two sinusoids that are played at the same pitch but at different phases But you can hear the "distance" between two pitches.

5 Phase Vocoders

As noted by, audio quality is .

Vertical and horizontal incoherence.

6 Algorithm Overview

6.1 STFT

Like almost all spectral algorithms, this one begins with a short-time Fourier transform (STFT) and ends with it's inverse (ISTFT). At the cost of reduced temporal resolution, the STFT gives us access to the spectral components of the signal in time.

The STFT performs a discrete time Fourier transforms on windows of size M of the original signal $x(n)$. These windows are separated by a hop size of R. The τth frame of the STFT is

$$X(\omega, \tau) = \mathrm{DTFT}(\tilde{x}(n, \tau) w(n)) \tag{1}$$

where

$$\tilde{x}(n, \tau) = x(n + \tau R) \tag{2}$$

is a buffer of the original signal, shifted R samples from the previous buffer and $w(n)$ is the synthesis window. $w(n)$ is defined to be nonzero for M samples, $n \in \left[-\frac{M-1}{2}, \frac{M-1}{2}\right]$, so only the M most recent samples of $x(n)$ need to be kept in memory.

After some computation, we produce frames $Y(\omega, \tau)$ which we want to use to synthesize the signal $y(n)$. We perform this ISTFT using the weighted overlap add method. As the name suggests, this method works by overlapping adjacent frames of the IDTFT, weighting them by an analysis window $f(n)$ and then adding them together. The weighting helps to suppress audible discontinuities at frame boundaries which prevents artifacts in highly nonlinear filters such as the one this paper describes.

$$\tilde{y}(n,\tau) = \mathrm{IDTFT}(Y(\omega,\tau)) \tag{3}$$

$$y(n) = \sum_{\tau} f(n-\tau R)\tilde{y}(n-\tau R,\tau) \tag{4}$$

Note that synthesizing a single sample of $y(n)$ requires overlapping $\frac{M}{R}$ frames which adds a latency of M samples between y and it's sources.

In the absence of spectral modifications, we can achieve perfect reconstruction of a signal transformed by the STFT and ISTFT given that the synthesis and analysis windows obey

$$\sum_{\tau} w(n-\tau R) f(n-\tau R) = 1,\ \forall n \in \mathbb{Z} \tag{5}$$

I chose to use the square root of the Hann window as both the analysis and frequency window which gives perfect reconstruction when $R = \frac{M}{2k}$ for $k \in \mathbb{Z}$.

$$w(n) = f(n) = \cos(\pi n/M) \tag{6}$$

For actual computation we replace, the DTFT and IDTFT with the FFT and IFFT respectively. The FFT has M bins $n \in \left[-\frac{M-1}{2}, \frac{M-1}{2}\right]$. The frequency of the nth bin is given by

$$\omega = \frac{f_s n}{M} \tag{7}$$

where f_s is sampling rate of the signal. Since the audio signal $x(n)$ is real, $X(\omega)$ is conjugate symmetric, so we only need to retain the positive bins $\left[0, \frac{M-1}{2}\right]$ of the FFT.

6.2 Spectrum Segmentation

As we In order to resolve the vertical incoherence — the phasing issues within the frame — my solution like others is to lock regions of the spectrum where phase relations are important. Rather than performing a transport over all bins, bins which are used to represent the same spectral information are lumped together.

TODO Many applications use simple peak finding, which is good in most appliccations but fails in certain settings. Instead I determine where to split the window based on the *reassigned frequency*. The reas

In order to fix the spec Techniques mention peak finding. However peak finding does not give a great sense of where to put boundaries. The reassigned frequency can be computed by:

$$\hat{\omega}(\omega) = \omega + \Im\left\{\frac{X_{\mathcal{D}}(\omega) \cdot X^*(\omega)}{|X(\omega)|^2}\right\} \tag{8}$$

where $X_{\mathcal{D}}$ is the spectrum of the x with the window $w_{\mathcal{D}}(n) = \frac{d}{dt}w(n)$:

$$X_{\mathcal{D}}(\omega) = \text{FFT}\left(w_{\mathcal{D}}(n)x(n)\right) \tag{9}$$

The derivative of our chosen window, the square root of the Hann window is:

$$w_{\mathcal{D}}(n) = -\frac{\pi f_s}{M}\sin\left(\frac{\pi n}{M}\right) \tag{10}$$

TODO: Plot of segmentation TODO: Formalize segmentation algorithm

This algorithm segments the spectrum into how at at most N segments $s \in S$. Each is has a center $c(s)$ and a set of indices $U(s)$. The mass of each segment is given by

$$\rho(s) = \frac{\sum_{n \in U(s)} |X(n)|}{\sum_n |X(n)|} \tag{11}$$

Note that

$$\sum_{s \in S} \rho(s) = 1 \tag{12}$$

6.3 Transport

After two audio signals have been segmented into S_0 and S_1 we then compute a transformation matrix $T \in \mathbb{R}^{S_0 \times S_1}$ whose entries represents the transfer of mass from one segment to the other. We want to construct this matrix so that the total movement of mass (or work) is minimized. We do not want to transport a negative mass or more mass than is available:

$$0 < T(s_0, s_1) \leq \max(\rho(s_0), \rho(s_1)) \tag{13}$$

Additionally, we want to constrain that all of the mass receives an assignment:

$$\rho(s_0) = \sum_{s_1 \in S_1} T(s_0, s_1) \; \forall s_0 \in S_0 \tag{14}$$

$$\rho(s_1) = \sum_{s_0 \in S_0} T(s_0, s_1) \; \forall s_1 \in S_1 \tag{15}$$

Given these constraints we want to find the assignment T that minimizes the 2-Wasserstein distance between the segments.

$$\min_T \sum_{\substack{s_0 \in S^0 \\ s_1 \in S^1}} T(s_0, s_1) |c(s_0) - c(s_1)|^2 \tag{16}$$

We compute such a T with algorithm 1.

Algorithm 1 Compute Transformation Matrix

$T \leftarrow 0$
$i, j \leftarrow 0$
$\rho_0 \leftarrow s_0^i$
$\rho_1 \leftarrow s_1^j$
while $i < |S_0|$ and $j < |S_1|$ **do**
  if $\rho_0 < \rho_1$ **then**
    $T(s_0^i, s_1^j) \leftarrow \rho_0$
    $i \leftarrow i + 1$
    $\rho_1 \leftarrow \rho_1 - \rho_0$
    $\rho_0 \leftarrow \rho(s_0^i)$
  else
    $T(s_0^i, s_1^j) \leftarrow \rho_1$
    $j \leftarrow j + 1$
    $\rho_0 \leftarrow \rho_0 - \rho_1$
    $\rho_1 \leftarrow \rho(s_1^i)$
  end if
end while

TODO: Plot of overlapping pieces
TODO: Proof - talk to Justin

6.4 Phase Accumulation

When we move a segment of the spectrum to a new pitch, the phases within that section rotate either slower or faster. This change does not make an audible difference within a window itself, but it can cause interference in the overlap between windows known as *horizontal incoherence.*

Include plot

One solution to this in time-stretching is to estimate the phase of the next frame based on its new instantaneous frequency. If a sinusoid with phase $\theta(\omega, \tau)$ is oscillating with instantaneous frequency $\hat{\omega}(\omega)$ over frame τ, we would expect the phase of the next frame to be

$$\theta(\omega, \tau + 1) = \theta(\omega, \tau) + \frac{R}{f_s}\hat{\omega}(\omega) \tag{17}$$

This model is good for signals which do not undergo rapid changes in frequency. However it does a poor job of dealing with transients in audio that can happen from sharp note attacks and percussive sounds. To cope with this, people employ reinitialization steps in frames where transients are detected, however this method is.

Fortunately however, we have at our disposal the phases of not one but two pitches. The center of will oscillate at $(1-k)\hat{\omega}(m_0) + \hat{\omega}_m^1$. Therefore we can interpolate.

For every bin i keep track of the accumulated phase θ_i^τ which we want to have the following properties:

$$\Theta_i^\tau \approx \sum_{t=0}^{\tau} \frac{R}{f_s}\hat{\omega}_i^t \tag{18}$$

$$\Theta_i^\tau = \theta_i^\tau \pmod{2\pi} \tag{19}$$

Therefore we want the synthesis phase of θ_i at the center of mass of segment m is

$$\theta_{c_m} = (1-k)\Theta_{c_m^0} + k\Theta_{c_m^1} \tag{20}$$

To preserve the vertical coherence between all bins within a singular we add $\theta'_{c_m} - \theta_{c_m}$ to all pitches

$$\theta'_i = \theta_i + \theta'_{c_m} - \theta_{c_m} \tag{21}$$

This way all the local relationships between phases are kept.

6.5 Resynthesis

Compute new center. Compute new phase for horizontal coherence Shift phases to preserve vertical coherence linearly interpolate between the resulting

Once the phases of each segment have been centered around the synthesis phase, we can synthesize a new spectrum:

for all n **do**
  $Y(n) \leftarrow 0$ ▷ Clear the output
end for

for $m_0 \in S_0$, $m_1 \in S_1$ **do**

  $\hat{n} \leftarrow (1-k)c(m_0) + kc(m_1)$ ▷ $\hat{n}$ is the new COM index
  $\hat{\Theta} \leftarrow (1-k)\Theta(c(m_0)) + k\Theta(c(m_1))$ ▷ θ' is the new COM phase

  for $n \in U(m_0)$ **do**

    $n' \leftarrow n + \hat{n} - c(m_0)$
    $\Theta' \leftarrow \Theta_0(n) + \hat{\Theta} - \Theta_0(c(m_0))$

    $Y(n') \leftarrow Y(n') + (1-k)\frac{T(m_0,m_1)}{\rho(m_0)}|X_0(n)|e^{i\theta'}$

  end for
end for

reentered by the phase accumulation, we can simply linearly interpolate the audio and add it to the spectrum. Since the

7 Implementation

I implemented the above algorithm in real-time as an audio effect in C++. The program listens to two streams of audio which can be sent to it from a any diginal audio workstation, like Ableton Live. It then produces a new stream of audio whose spectrum is transported between the two according to a MIDI value. The program depends on FFTW (Fastest Fourier Transform in the West) for Fourier transforms, and the open source libraries PortAudio and PortMidi for audio and MIDI handling respectively.

I work with a window size of 4096 samples and a sampling rate of 10 milliseconds. This is audible.

My code is available at `https://github.com/sportdeath/Vocoder`

A video of my implimentation is available at.

8 Conclusion

As seen in the implimentation video, the sound quality is good for many types of signals.

One of the most noticeable artifacts that still exists is the interference that results from a single pitch being mapped to many places. As the note splits and moves to new locations it begins to interfere with itself. This can cause a drop in volume at the very start or end of a transport. This could be fixed by requiring that the mapping be fixed.

There are however sounds that the effect does not work for. If we try to transport to audio which has rapid changes in pitch, the source material deteriorates quickly because the transport map is not consistent. In certain settings however this can be used musically. If for example the we are transporting between two sounds with relatively constant pitch.

But overall this the sound quality is good for many sounds. It has the potential to be used by DJs looking for interesting ways to mix between songs, and musicians looking to create new sounds.

If one of the sources is time varying the source material quickly deteriorates. Great for constant sounds.

9 References

Fundemental theory

that phase vocoder one

1d transport paper?

STFT

weighted overlap add

fftw

portaudio

portmidi

cite justin solomon