# An algorithm for morphing audio

Berge, M. van den

# Universiteit Leiden
# Opleiding Informatica

An algorithm for

morphing audio


Name: Michiel van den Berge

Date: 17/08/2015

1st supervisor: Dhr. O.D. Biesel (MI)
2nd supervisor: Dhr. E.M. Bakker (LIACS)


BACHELOR THESIS


Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

**Abstract**

In this paper we look at different ways of morphing audio and introduce a novel strategy for morphing as an attempt to solve difficulties encountered in partial based models. After proving that this is a well-defined strategy we try to characterize the workings and shortcomings of the new morph by looking at various types of input. We conclude that the morph produces sounds at unwanted frequencies and often morphs in a way that produces inharmonic sounds, making it unsuited for musical and other harmonic audio, and that the morph provides interesting results on sounds consisting mainly of noise.

# Contents

# 1 Introduction

This paper focuses on various strategies for morphing audio. A morph is a 'continuous' transformation of two (or more) inputs: given any two inputs it will produce a range of things that are 'in-between' them, slowly transforming from one to the other. It should be noted that there are often many possible ways to construct a morph; we focus on constructing a morph that yields interesting results throughout the morphing process.

A common approach is to describe the input in terms of certain characteristics or features. Once a set of features has been chosen that describes the input as completely as possible one can construct a morph by altering these features. Ideally the chosen features are meaningful and apparent to the listener: directly interpolating such features would create a morph that smoothly changes sounds. We will look at the sets of features used in previous research and choose the amplitude spectrum as our main feature. Then we will introduce a novel way of interpolating such spectra by taking their integrals and weighing them using the geometric mean. Finally we discuss a method to create a new sound file from such an interpolation using phase information.

First we will look at the Fourier transform as it will be needed for working with and describing audio. Then we look at various state of the art techniques and introduce the concept of our morph. In section 4 we give an algorithm for computing this morph while checking it is well-defined. Section 5 contains examples of various morphed audio files and describes the characteristics of the morph. These results are summarized in section 6 along with questions for further research.

# 2 Fourier transform

Our morph is meant primarily for musical sounds. These sounds usually have a clearly identifiable pitch; there is a fundamental frequency which represents the pitch of the note and a range of other frequencies that help shape the sound (usually called overtones) that differ for each instrument. When morphing musical sounds it is very helpful to see which frequencies a sound consists of so that the pitch and overtones become apparent. In this section we will use the Fourier transform to easily access the frequency information of a sound.

## 2.1 Continuous Fourier transform

We start with some basic definitions of our audio input. A sound can be seen as a periodic vibration which is usually transmitted through the air. It is possible to record sounds by measuring the variations in air pressure and converting these to an electronic signal. Since we need an electronic signal in order to work with computers we take this as the definition for our audio input.

**Definition 2.1.** An *audio signal* of length $T$ is a continuous function $[0, T] \to \mathbb{R}$.

To access the frequency information we can embed such a signal in a vector space with a basis of simple frequencies.

**Definition 2.2.** The space of *square integrable functions* on an interval $[a, b]$ is the set $L^2[a, b] = \{f \colon [a, b] \to \mathbb{C} \mid \int_a^b |f(t)|^2 dt < \infty\}$.

We can define pointwise addition and multiplication by constants to create a vector space. Since audio signals are continuous $L^2[0,T]$ contains all audio signals of length $T$. We can define an inner product on $L^2[a,b]$ by setting

$$\langle f, g\rangle = \int_a^b f(t)\overline{g(t)}dt.$$

This leads to the following nice result:

**Proposition 2.3.** *Let $L^2[a,b]$ and set $L = b-a$. Then the functions $\{e^{2\pi ikt/L}\}_{k\in\mathbb{Z}}$ form an orthogonal basis of $L^2[a,b]$ and the normalized functions $\{\frac{1}{\sqrt{L}}e^{2\pi ikt/L}\}_{k\in\mathbb{Z}}$ form an orthonormal basis.*

For a proof we refer to [1]. The orthogonality gives us an easy way to calculate the coördinates with respect to the basis.

**Definition 2.4.** Let $f$ be an audio signal of length $T$. The *Fourier transform* $\hat{f} : \mathbb{Z} \to \mathbb{C}$ of $f$ is the function

$$\hat{f}(k) = \frac{1}{T}\cdot\langle f, e^{2\pi ikt/T}\rangle = \frac{1}{T}\cdot\int_0^T f(t)e^{-2\pi ikt/T}dt$$

which for each $k \in \mathbb{Z}$ gives the coördinate of $f$ with respect to the $k$th basis vector.

It is important to note that we can express our audio signal in terms of its Fourier transform by summing over all basis vectors and their coëfficients. For all $x \in [0,T]$ we have

$$f(x) = \sum_{k=-\infty}^{\infty} \hat{f}(k)\cdot e^{2\pi ikx/T}$$

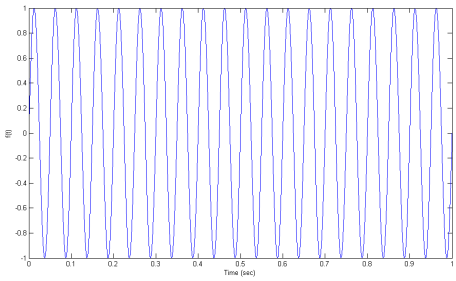
(do note that this holds only because we demanded our audio signal to be continuous[2]; for a general $L^2$-function this only holds almost everywhere). In particular, we can easily reconstruct an audio signal from its Fourier transform. The signal is now written as a sum of complex functions, yet the signal itself is real-valued. This leads to some constraints on the Fourier transform.

**Lemma 2.5.** *Let $f$ be an audio signal of length $T$ and $\hat{f}$ its transform. Then for all $k \in \mathbb{Z}$ we have $\hat{f}(k) = \overline{\hat{f}(-k)}$ and $\hat{f}(0) \in \mathbb{R}$.*

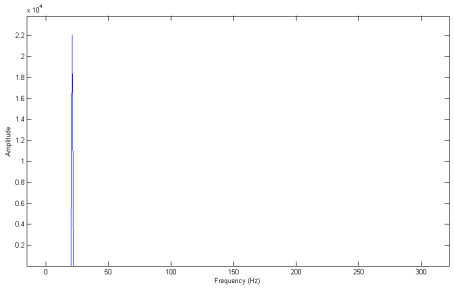
*Proof.* We write $f(x) = \sum_{k=-\infty}^{\infty} \hat{f}(k)\cdot e^{2\pi ikx/T}$ for any $x \in [0,T]$. Since $f(x)$ is real we have

$$\begin{aligned} f(x) &= \sum_{k=-\infty}^{\infty} \hat{f}(k)\cdot e^{2\pi ikx/T} \\ &= \overline{f(x)} = \sum_{k=-\infty}^{\infty} \overline{\hat{f}(k)}\cdot e^{-2\pi ikx/T} \\ &= \sum_{k=-\infty}^{\infty} \overline{\hat{f}(-k)}\cdot e^{2\pi ikx/T}. \end{aligned}$$

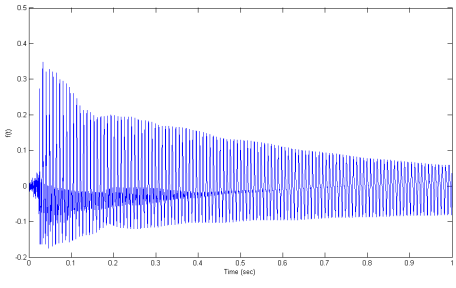
Since the coëfficients of $f$ with respect to our basis are unique it follows that $\hat{f}(k) = \overline{\hat{f}(-k)}$ for all $k \in \mathbb{Z}$. □



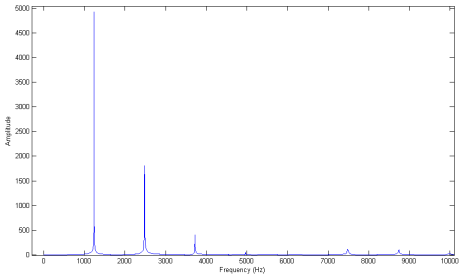
(a) simple audio signal



(b) corresponding amplitude spectrum



(c) audio signal of a musical instrument



(d) corresponding amplitude spectrum

Figure 1: Two audio signals and their amplitude spectrum.

Now we can take a look at the meaning of the transform. Let $k \in \mathbb{Z}_{\geq 1}$ and write $\hat{f}(k) = r(k)\cos(\phi(k)) + ir(k)\sin(\phi(k))$. Then

$$\begin{aligned}
f(x) &= \sum_{k=-\infty}^{\infty} \hat{f}(k) \cdot e^{2\pi ikx/T} \\
&= \hat{f}(0) + \sum_{k=1}^{\infty} \left( \hat{f}(k) \cdot e^{2\pi ikx/T} + \hat{f}(-k) \cdot e^{-2\pi ikx/T} \right) \\
&= \hat{f}(0) + \sum_{k=1}^{\infty} \left( (\hat{f}(k) + \hat{f}(-k)) \cos(2\pi kx/T) + i(\hat{f}(k) - \hat{f}(-k)) \sin(2\pi kx/T) \right) \\
&= \hat{f}(0) + \sum_{k=1}^{\infty} \left( 2r(k)\cos(\phi(k))\cos(2\pi kx/T) - 2r(k)\sin(\phi(k))\sin(2\pi kx/T) \right) \\
&= \hat{f}(0) + \sum_{k=1}^{\infty} 2r(k)\cos(2\pi kx/T + \phi(k)).
\end{aligned}$$

The Fourier transform thus gives a decomposition of an audio signal in basic sine waves with amplitude $2r$ and phase shift $\phi$. An audio signal consisting of a single sine wave is perceived as a single note of which the pitch depends on the frequency and the loudness depends on the amplitude of the sine. The modulus of $\hat{f}(k)$ tells us the amplitude of the sine wave with frequency $\frac{k}{T}$, which corresponds to the loudness of a note with the same frequency: it tells us how much that pitch is present in the original audio signal. A plot of $|\hat{f}|$ is called an *amplitude spectrum*.

Examples of amplitude spectra can be seen in figure 1. Figure 1a shows an audio signal of a simple sine wave with a frequency of 20 Hz. Without any further analysis the Fourier

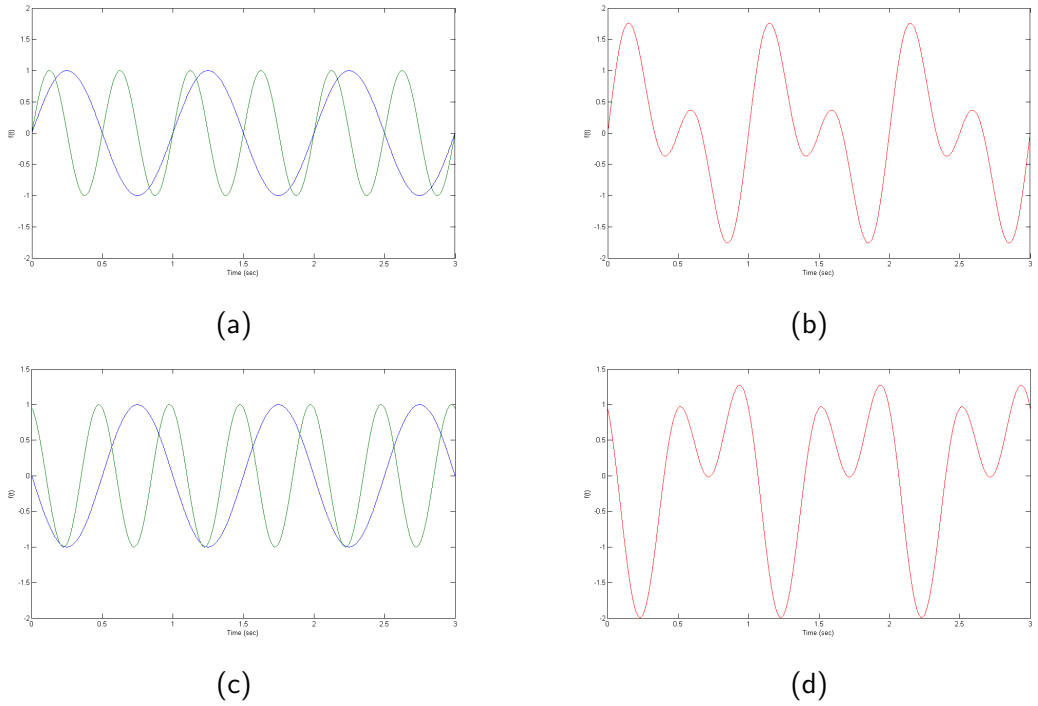



Figure 2: 2a and 2c both contain a wave of 1Hz and a wave of 2Hz but in different phase configurations. As a result the sums of the two waves are shaped differently.

transform indeed shows that the signal consists only of a sine of 20 Hz. Figure 1c contains the signal of a musical instrument playing a single note. Here it is not as easy to infer information directly from the signal, but the amplitude spectrum provides clear information on the pitch and overtones of the sound.

It should be noted that the peaks in figure 1d are not as narrow as the peak in figure 1b. This is because the Fourier transform decomposes the original sound in waves with frequencies that are multiples of $\frac{1}{T}$. If a sound contains frequencies that are not multiples of $\frac{1}{T}$ some inaccuracies will show. The amplitude of such a frequency is in some way split over the nearby available frequencies. For this work it is not hugely important how exactly this happens so we will not explore this further.

The phase information encoded in the argument of $\hat{f}(k)$ doesn't have as clear a meaning as the amplitude information. The phases of multiple frequencies dictate how these frequencies interact, i.e. where they boost and attenuate each other. This information does not impact the pitch of the audio signal but rather the shape it takes, like temporary increases in loudness or in the case of musical instruments often a sudden increase followed by a gradual lowering of volume when a note is struck. Figure 2 contains an example. In 2a the two waves both have their phase set to zero; in 2b the phase of the lower frequency is set to $\pi$ and the phase of the higher frequency is set to $\frac{2\pi}{3}$. As a result the sums of these waves behave differently.

## 2.2 Discrete Fourier transform

The audio signals we have worked with so far are continuous functions from a subset of $\mathbb{R}$. While such continuous signals are generated during the recording of audio a lot of data is lost when these signals are stored, since only finitely many points can be stored on a computer.

When a computer or recording device receives an audio signal as input it will measure that signal at regular intervals. The measurements are called samples and the process is called sampling.

**Definition 2.6.** A *sampled audio signal* of length $T$ with $N$ samples is a function $[0,T] \supset \{\frac{1}{N}T, \frac{2}{N}T, \ldots, \frac{N-1}{N}T, T\} \to \mathbb{R}$.

*Notation.* Let $f$ be a sampled audio signal with $N$ samples. For $1 \leq n \leq N$ we denote the $n$th sample as $f[n] = f(\frac{n}{N}T)$.

For fixed $N$ we can think of a sampled audio signal as a vector of length $N$. Analogous to the continuous case we can form a basis of simple frequencies. In the vector representation these take the form $(e^{2\pi ik1/N}, \ldots, e^{2\pi ik(N-1)/N}, e^{2\pi ikN/N})$[3]. This leads to the following analogue of definition 2.4:

**Definition 2.7.** Let $f$ be a sampled audio signal of length $T$ with $N$ samples. The *Fourier transform* $\hat{f} : \mathbb{Z} \to \mathbb{C}$ of $f$ is the function

$$\hat{f}(k) = \frac{1}{N} \cdot \langle f, e^{2\pi ikn/N} \rangle = \frac{1}{N} \cdot \sum_{n=1}^{N} f[n] e^{-2\pi ikn/N}$$

which for each $k \in \mathbb{Z}$ gives the coördinate of $f$ with respect to the $k$th basis vector.

One can show that again $\hat{f}(k) = \overline{\hat{f}(-k)}$ for all $k \in \mathbb{Z}$. As it turns out the Fourier transform of sampled signals have another rather helpful property.

**Proposition 2.8.** *Let $f$ be a sampled audio signal of length $T$ with $N$ samples. Then the Fourier transform $\hat{f}$ is periodic with period $N$.*

*Proof.* This follows straight from the definition. For any $k \in \mathbb{Z}$ we have

$$\hat{f}(k+N) = \frac{1}{N} \cdot \sum_{n=1}^{N} f[n] e^{-2\pi i(k+N)n/N} = \frac{1}{N} \cdot \sum_{n=1}^{N} f[n] e^{-2\pi ikn/N} e^{-2\pi in} = \hat{f}(k).$$

□

In the process of sampling a lot of data is lost. As a consequence the Fourier transform also yields less data. In live purposes, where small chunks of audio have to be processed as fast as possible to minimize any delay, this creates a challenge where taking too small an audio signal means the Fourier transform does not contain any useful data. On the whole though it is extremely convenient that the transform of a finite signal is itself again finite. Just like in the continuous case we can express an audio signal in terms of its Fourier transform[3]:

$$f(n) = \sum_{k=1}^{N} \hat{f}(k) e^{2\pi ink/N}.$$

So the Fourier transform of a sampled signal is very convenient to work with. For a signal of $N$ samples the transform is completely determined by $\frac{N}{2} + 1$ values and it is easy to recover the original signal. Working straight from these equations calculating an (inverse) transform would have time complexity $O(N^2)$. A wide selection of algorithms are available that do these transforms in $O(N \log_2 N)$, the first of which dates to 1965[4]. Any such algorithm with complexity $O(N \log_2 N)$ is called a Fast Fourier transform.

# 3 Morphing strategies

A morph is a way of transforming two inputs in a continuous fashion. In the case of audio a morph would be a transformation between two arbitrary audio signals. So given two audio signals the morph gives rise to a set of new signals that change from one sound to the other. As mentioned previously there are often myriad ways to do this, some more complicated than others. It may be helpful to visualize a space of audio signals and see the morph as a path from one to the other. Some paths do not contain any interesting sounds, some might contain sound that is barely audible; ideally a morph can provide an interesting path for a wide variety of signals.

The path you choose is highly dependent on which characteristics of the input you would like to control. For example, musical vibrato is often lost when morphing. When taking sounds with a soft vibrato and a harsh vibrato as input a morph can produce 'in-between' sounds that do not have any vibrato at all. Research has been conducted to create morphs that specifically respect the vibrato [5], but this places such a demand on the path you choose that you lose control over some other feature the original morph might have respected. In brief, different morphs emphasize different aspects of your input and are thus suitable for different goals. Deciding on a morphing strategy is thus closely related to the set of features you choose to emphasize. In this section we take a look at previous morphs and the features they emphasize and introduce a new morph.

## 3.1 Previous work

For a nicely detailed summary on fairly recent work we refer to the thesis of Regueiro[5] which provides plenty of further references. Most features are either derived from the signal directly or from its transform. We refer to features derived directly as features in the time domain, and to features derived from the transform as features in the frequency domain. An example of a feature in the time domain would be the amplitude over time, or rather the amplitude envelope. This relates to time-based events such as the beginning or attack of a musical note.

When working in the frequency domain there is the issue of noise. Due to a variety of reasons noise is often present which in the frequency domain manifests itself as energy that is present across all frequencies. That is, noise can be seen as random interference in every frequency. This makes it more difficult to manipulate the frequency domain both from a computational standpoint (as treating all frequencies is often computationally expensive) and because it can be unclear which frequencies contribute something meaningful to the sound as opposed to frequencies which contain nothing but noise. Multiple solutions have been proposed to alleviate this. In the mid-eighties a model called Spectral modeling synthesis, or SMS, was proposed[6]. The core idea was to see the frequency spectrum as the sum of a deterministic part and a stochastic part; the stochastic part represents the noise and the deterministic part consists only of clear peaks called *partials*. The harmonic content of a sound would reside only in these partials and could as such be easily altered while the noise could be processed seperately.

A related model developed in the early 2000's is Loris[7]. This model also works with partials but does not contain a seperate layer for noise. Instead each partial contains a parameter for noise energy. When a partial contains no noise it is still a pure sine wave of one clear frequency, visible in a spectrum as one clear peak. As more noise is added the resulting function will look like a sum of sine waves of frequencies close to the frequency of the partial,

causing a 'spill' of frequencies which looks like the original frequency is smeared out over the spectrum.

The approach of using partials is quite commonly used in audio morphing. Features in the frequency domain can almost always be expressed in terms of partials (an exception would be vibrato which concerns small fluctiations of frequencies). Most of the work is then to analyse a sound and find a representation in terms of partials, which can be quite difficult. Especially in noisy sounds it is not necessarily clear what might be a partial or particularily violent noise. When morphing sounds there is a problem of alignment. When the input sounds are similar (e.g. of the same instrument) the sets of partials will most likely resemble each other such that a pairwise identification can be made. However, often the input sounds will not be similar in which case there might be no evident way of aligning the partials - the number of partials will already often differ. The method in which partials are matched will then have a tremendous impact on the morph.

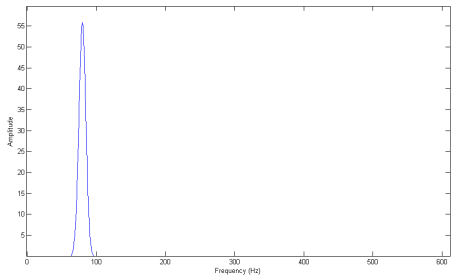
There are other approaches to analysing features in the frequency domain. Efforts have been made to create models that hardly use Fourier transforms altogether[8], and there is documentation on using genetic algorithms to generate new spectral envelopes[9]. However, models using partials seem to remain the standard.

## 3.2 A new strategy

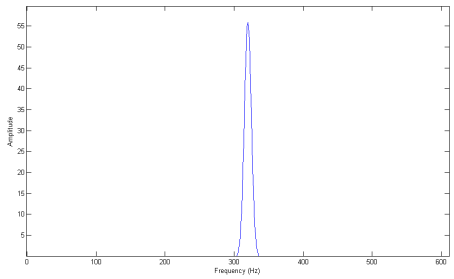
As mentioned in the last section, working with partials brings problems of both finding and aligning partials. We propose a novel way of morphing two spectra without using partials. The main idea is to take two positive real-valued functions $f$ and $g$ (such as the amplitude for each frequency) and focus instead on morphing their integrals $F$ and $G$. Both integrals can be taken and scaled on both axes such that they are comparable, for example by meeting at $(0, 0)$ and $(1, 1)$. Since we took our functions to be positive their integrals are non-decreasing which makes them easier to compare. We can create a new function $H$ by finding $x, y$ such that $F(x) = G(y)$ and setting $H(x^{(1-p)} \cdot y^p) = F(x) = G(y)$ for some parameter $p \in [0, 1]$, where $x^{(1-p)} \cdot y^p$ is the weighted geometric mean of $x$ and $y$. Lastly we can find a new function $h$ by differentiating. Please see figure 3 for an illustration.

The reasoning for using integrals is that these are more easily manipulated mathematically while retaining all information present in the original function. The interpolation used when creating $H$ is meant to function like a partial model but without the rigidity of defining and finding partials. Any clear peak will be visible in the integral as a steep incline. Such an incline is moved smoothly along the $x$-axis during morphing which causes the corresponding peak to smoothly slide along the frequency axis as seen in figure 3. Using integrals and this interpolation method allows the peaks to move freely without needing identification.

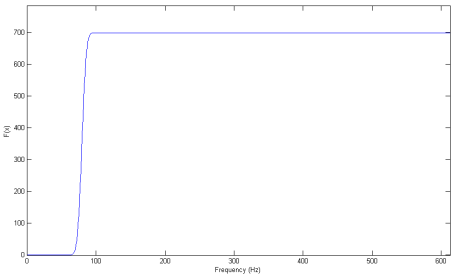
Interesting things will happen when the number of partials and their shapes do not match. When one signal contains a partial that cannot be matched to another, a new partial of zero amplitude is created in the other signal in or near the same frequency (the frequency might be altered slightly to better fit the harmonic content of the other sound). This causes the previously unmatched partial to slowly dissappear when morphing as its amplitude nears zero; conversely, starting from the other sound a new partial will emerge from nothing. In our approach such an unmatched partial is informally matched to some other and will naturally move towards it. Take as an example a sound with one partial morphed to a sound with two partials. The one partial will slowly grow wider and split in two partials that will move to their own frequency. Morphing in the other direction the two partials will move toward each other



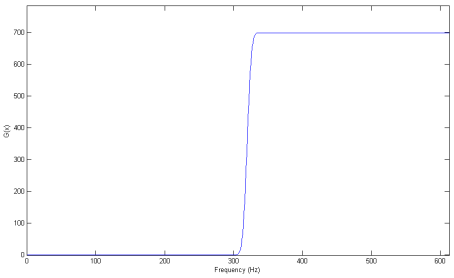

(a) f



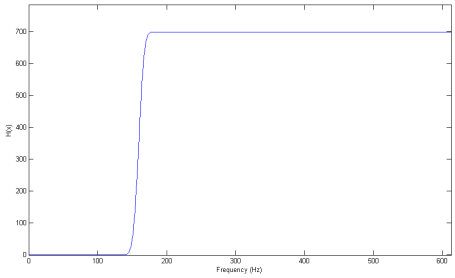

(b) g



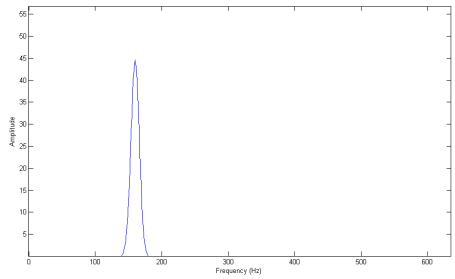

(c) F




(d) G




(e) H

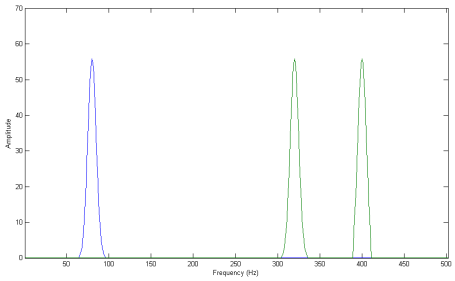



(f) h

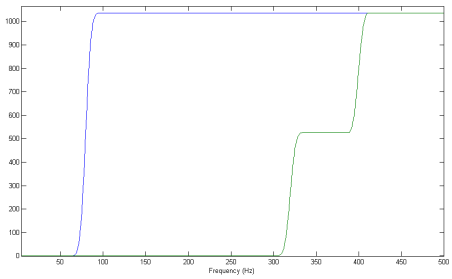
Figure 3: The morphing process. We start with two functions $f$ and $g$ and create their integrals $F$ and $G$. From these a function $H$ is created which yields a function $h$ by differentiating.

and merge. An example can be found in figure 4.

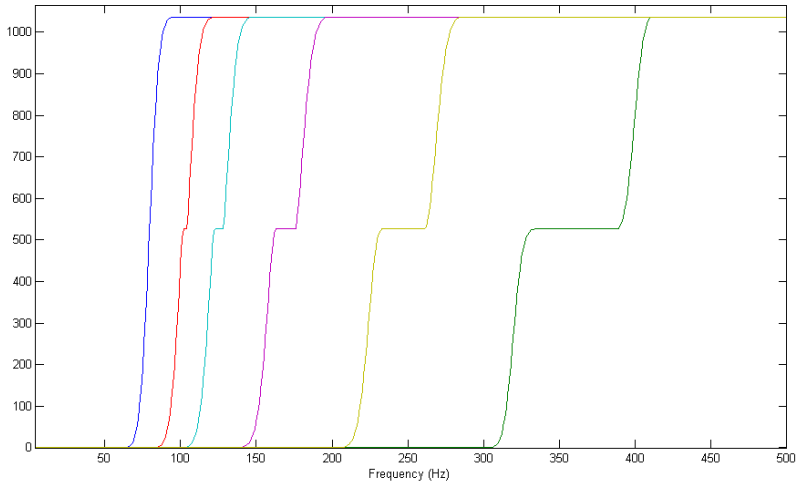
Lastly there is the fact that we used a geometric mean when defining $H$. The geometric mean works quite well in musical sounds since it preserves the ratio between different frequencies, and these ratios are extremely important in our perception of sound. As an example, take one sound with a frequency of 50Hz and a frequency of 75Hz, and one sound with a frequency of 100Hz and a frequency of 150Hz. Both sounds contain two frequencies with ratio $2 : 3$.



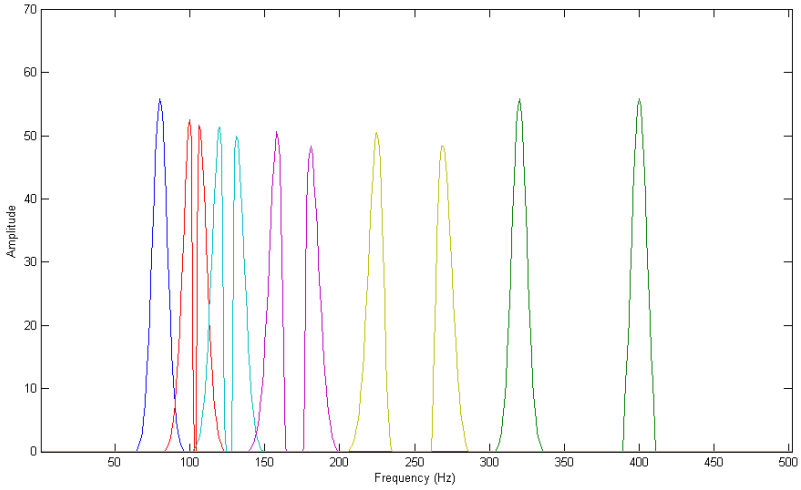

(a) The two spectra taken as input




(b) The two corresponding integrals




(c) A number of interpolated functions, calculated with $p = 0.17, p = 0.3, p = 0.5$ and $p = 0.75$




(d) The resulting spectra

Figure 4: Starting with two differently shaped spectra the morph smoothly changes the shape of the spectrum.

When morphing we would like the intermediate sounds to contain two frequencies with this ratio as well. Taking the standard arithmetic mean we would find a signal with frequencies

75Hz and 125Hz halfway through; the geometric mean gives frequencies of roughly 70Hz and 106Hz, preserving the $2:3$ ratio. For this reason we use a geometric mean for $H$.

# 4 Calculations

Last section we informally discussed previous morphing strategies and outlined the approach we will take. In this section we will define our morph and prove it is well-defined. We also present an algorithm for computing the morph.

Before we start this chapter there is an important matter of notation to be resolved. The algorithm was implemented in MATLAB[10] which starts array indices at 1. As a result all indices had to be shifted, e.g. given some Fourier transform $\hat{f}$ one would need to ask MATLAB for the first value to obtain $\hat{f}(0)$, ask for the second value to obtain $\hat{f}(1)$ and so forth. As it so happens this convention simplifies notation for the upcoming math somewhat so we have kept it for the following chapters: for $1 \leq n \leq N$ we denote $\hat{f}[n] = \hat{f}(n-1)$. Lemma 2.5 then states that $\hat{f}[1] \in \mathbb{R}$ and

$$\hat{f}[n] = \hat{f}(n-1) = \overline{\hat{f}(N+1-n)} = \overline{\hat{f}[N+2-n]}$$

for $2 \leq n \leq N$.

## 4.1 Defining the morph

First we will make our previous description more rigorous. As input we take two sampled audio signals $f$ and $g$, both of length $T$ with $N$ samples, and some percentage $p \in [0,1]$. The output will be a sampled audio signal (also of length $T$ with $N$ samples) that is produced by stopping the morph after $p$ percent when morphing from $f$ to $g$. So for $p = 0$ the output should be $f$, for $p = 1$ the output should be $g$, for $p = 0.2$ the output should be something close to $f$ that slightly resembles $g$ and so forth.

First we need to use $f$ and $g$ to construct two continuous functions $F, G$ that are non-decreasing and map to the same image. We will discuss our method for this later this section and first focus on using these two functions to construct $H$.

**Theorem 4.1.** *Let $F, G : [1, N] \to [0, M]$ be continuous and non-decreasing. Then there exists a unique function $H : [1, N] \to [0, M]$ such that for all $x, y \in [1, N]$*

$$F(x) = G(y) \Rightarrow H(x^{(1-p)} \cdot y^p) = F(x) = G(y).$$

*Proof.* First we prove that $H$ exists for all values of $p$. For $p \in \{0, 1\}$ this is obvious since $F$ and $G$ share the same image, so $H$ will be either $F$ or $G$ itself; so we can take $p \in (0, 1)$. Let $n \in [1, N]$: we prove that $H(n)$ exists. Should $F(n) = G(n)$ then we are done, so $F(n) > G(n)$ or $F(n) < G(n)$. First we consider $F(n) > G(n)$.

Let $x(t) = (1-t)n + t$ and $y(t) = (n \cdot x(t)^{p-1})^{\frac{1}{p}}$ for $t \in [0, 1]$. Then for all $t \in [0, 1]$ we have $x(t)^{(1-p)} \cdot y(t)^p = n$. We can now consider $F(x(t))$ and $G(y(t))$ as functions of $t$ where $F(x(0)) > G(y(0))$ and try to find a value of $t$ such that $F(x(t)) < G(y(t))$; then by the intermediate value theorem we obtain a value $t^*$ such that $F(x(t^*)) = G(y(t^*))$. For $t = 1$ we have $F(x(t)) = 1$, the minimum of the function, but $G(y(1))$ is not necessarily defined since $y(1)$ might be larger than $N$. In such a case we can take $t^* \in (0, 1)$ such that $y(t^*) = N$ and thus $G(y(t^*)) = M$, which still yields $F(x(t^*)) \leq G(y(t^*))$.

This proves that for $F(n) > G(n)$ we can find a value for $H(n)$. In the case that $F(n) < G(n)$ we can take a similar approach by setting $y(t) = (1-t)n+1$ and $x(t) = (n \cdot y(t)^{-p})^{\frac{1}{(1-p)}}$. So for any $n \in (1, N)$ we can find a value for $H(n)$: all that remains is to show that these values are unique. Let $x, y, x', y' \in [1, N]$ such that $F(x) = G(y)$, $F(x') = G(y')$ and $x^{(1-p)} \cdot y^p = x'^{(1-p)} \cdot y'^p$. If $x \neq x'$(and so $y \neq y'$) assume without loss of generality that $x > x'$ and $y < y'$. Since $F$ and $G$ are non-decreasing we have

$$F(x') \leq F(x) = G(y) \leq G(y')$$

and thus we have $F(x) = F(x') = G(y) = G(y')$. We have proven that for all $p \in [0, 1]$ we can find a well-defined function $H$ that is completely determined by $F$ and $G$: hence $H$ is also unique. □

*Notation.* Given two continuous and non-decreasing functions $F, G : [1, N] \to [0, M]$ we call $H$ their *direct interpolation*.

**Corollary 4.2.** *Let $F, G : [1, N] \to [0, M]$ be continuous and non-decreasing, and let $H$ be their direct interpolation. Then $H$ is non-decreasing.*

*Proof.* Let $n, n' \in [1, N]$ such that $H(n) > H(n')$ and let $x, y, x', y' \in [1, N]$ such that $F(x) = G(y) = H(x^{(1-p)} \cdot y^p)$, $F(x') = G(y') = H(x'^{(1-p)} \cdot y'^p)$, $x^{(1-p)} \cdot y^p = n$ and $x'^{(1-p)} \cdot y'^p = n'$. Then from $H(n) > H(n')$ it follows that $F(x) > F(x')$ and $G(y) > G(y')$, so $x \geq x'$ and $y \geq y'$. Then we also get $n = x^{(1-p)} \cdot y^p \geq x'^{(1-p)} \cdot y'^p = n'$ which proves that $H$ is non-decreasing. □

Now it is clear how we can obtain an interpolation $H$ from $F$ and $G$. We would like to construct a new audio signal $h$ using $H$; for doing so it is important to decide how to construct $F$ and $G$ from our input signals such that we can reverse the process. Starting from our input signals we will create some intermediate functions that reflect the steps described in the previous section: creating positive real-valued functions, integrating, scaling as well as making the resulting function continuous.

Given our input signals $f$ and $g$ of $N$ samples we take Fourier transforms $\hat{f}$ and $\hat{g}$. To create positive real-valued functions two options seem apparent. We could take the amplitude spectrum, meaning the modulus of the Fourier transform; or we could take the modulus squared. This would represent the total energy present in each frequency (this is due to Parseval's identity[11] which states a relation to the squares of the Fourier coefficients and the square of the original signal). The impact of this choice will be discussed next chapter.

Next we want to take the integral of our positive function. Since integrating over sampled functions makes little sense we take sums instead, and get

$$F_1(n) = \sum_{k=2}^{n} |\hat{f}[k]| \text{ or } F_1(n) = \sum_{k=2}^{n} |\hat{f}[k]|^2$$

as our function for $2 \leq n \leq N$, $x \in \mathbb{Z}$. We do need to ensure that $F$ and $G$ will have the same image. We do this by demanding that $F_1(1) = G_1(1) = 0$ and setting

$$F_2(n) = F_1(n) \cdot \frac{1}{2} \frac{F_1(N) + G_1(N)}{F_1(N)}$$

(analogous for $G$) such that both functions obtain the same maximum value, say $M$. Now, these functions are still only defined on $N$ points. We want to find $x, y$ such that $F(x) = G(y)$

but for these functions such points do not need to exist; so we set $F$ and $G$ to be the continuous functions obtained by linearly interpolating $F_2$ and $G_2$.

Now we can take the direct interpolation $H$ of $F$ and $G$ and reverse these steps. First we need to undo the scaling we applied on $F$ and $G$, so say

$$H_1(n) = H(n) \cdot \frac{(1-p) \cdot F_1(N) + p \cdot G_1(N)}{\frac{1}{2} \cdot (F_1(N) + G_1(N))}$$

where the maximum value of $H_1$ is interpolated linearly between the maxima of $F_1$ and $G_1$ depending on $p$. Next we can set $|\hat{h}[k]| = H_1(k) - H_1(k-1)$ or $|\hat{h}[k]| = \sqrt{H_1(k) - H_1(k-1)}$ for $2 \leq k \leq N$, depending on our earlier choice. The values $\hat{f}[1]$ and $\hat{g}[1]$ were not encoded in $F$ and $G$ so we manually set $\hat{h}[1] = (1-p) \cdot \hat{f}[1] + p \cdot \hat{g}[1]$.

The rather apparent problem at this point is that we only constructed the absolute value of $\hat{h}$ and have no way to determine the function itself which is needed to compute the new audio signal. $F$ and $G$ only contain information on the modulus of $\hat{f}$ and $\hat{g}$. To finish constructing $\hat{h}$ we also need the phase information. For this purpose we construct new functions $F_{phase}, G_{phase} : [1, N] \to (-\pi, \pi]$ that at each integer $1 \leq n \leq N$ contain the phase information (or rather the argument) of either $\hat{f}[n]$ or $\hat{g}[n]$. Between integers the phase can be interpolated (we will briefly visit the effect of the interpolation method next chapter). These functions can be used to create $H_{phase}$: when there are $x, y$ such that $H(x^{(1-p)} \cdot y^p) = F(x) = G(y)$ we set $H_{phase}(x^{(1-p)} \cdot y^p)$ to be some interpolation of $F_{phase}(x)$ and $G_{phase}(y)$ (we will also visit the effect of this interpolation method next chapter). Then finally the phase contained in $H_{phase}(k)$ is taken as the argument of $|\hat{h}[k]|$ and we obtain a complete Fourier transform $\hat{h}$. Taking the inverse transform yields the desired output $h$.

## 4.2 Well-definedness of $\hat{h}$

In the previous section we have defined our morph using Fourier transforms and various functions derived from these transforms. We started with signals of $N$ samples and treated the transforms as if they had $N$ distinct points of data, since these transforms are $N$-periodic. In this section we use the identity of lemma 2.5, manifesting as $\hat{f}[n] = \overline{\hat{f}[N+2-n]}$ for $2 \leq n \leq N$, to prove the derived functions $F$ and $G$ have a similar kind of symmetric behaviour. This allows us to know these functions by only computing $\frac{N}{2}+1$ points of them which cuts out some computational effort. Lastly we enforce such a condition on $H$ thus ensuring $\hat{h}$ satisfies the identity of lemma 2.5 and that its inverse transform $h$ is indeed real-valued.

**Proposition 4.3.** *Let $f$ be an audio signal of $N$ samples and construct $F : [1, N] \to [0, M]$ as in the previous section. Then for all $r \in [1, N]$ we have $F(r) + F(N+1-r) = M$.*

*Proof.* First we will prove that for all $1 \leq n \leq N$ we have $F_1(n) + F_1(N+1-n) = F_1(N)$. For $n = 1$ or $n = N$ the statement is trivial. Otherwise using that $|\hat{f}[n]| = |\hat{f}[N+2-n]|$ we

have

$$\begin{aligned} F_1(n) + F_1(N+1-n) &= \sum_{k=2}^{n} |\hat{f}[k]| + \sum_{k=2}^{N+1-n} |\hat{f}[k]| \\ &= \sum_{k=2}^{n} |\hat{f}[k]| + \sum_{k=n+1}^{N} |\hat{f}[k]| \\ &= \sum_{k=2}^{n} |\hat{f}[k]| = F_1(N). \end{aligned}$$

(this works the same when setting $F_1(n) = \sum_{k=2}^{n} |\hat{f}[k]|^2$.) The same property holds for $F_2$ as it is a scaled version of $F_1$. We use this to prove our property of $F$. Again for $r = 1$ or $r = N$ this is trivial, so assume $r \in (1, N)$. Write $r = n + c$ with $n \in \mathbb{Z}$ and $c \in [0, 1)$. $F$ is the linear interpolation of points of $F_2$ so $F(r) = F(n + c) = (1 - c)F_2(n) + cF_2(n + 1)$. This gives

$$F(r) + F(N + 1 - r) = (1 - c)F_2(n) + cF_2(n + 1) + cF_2(N - n) + (1 - c)F_2(N + 1 - n).$$

Since $F_2(n) + F_2(N + 1 - n) = F_2(N) = M$ we can write

$$cF_2(N - n) + (1 - c)F_2(N + 1 - n) = c(M - F_2(n + 1)) + (1 - c)(M - F_2(n))$$

which yields

$$\begin{aligned} & F(r) + F(N + 1 - r) \\ = \; & (1 - c)F_2(n) + cF_2(n + 1) + cF_2(N - n) + (1 - c)F_2(N + 1 - n) \\ = \; & (1 - c)F_2(n) + cF_2(n + 1) + c(M - F_2(n + 1)) + (1 - c)(M - F_2(n)) \\ = \; & (1 - c)F_2(n) - (1 - c)F_2(n) + cF_2(n + 1) - cF_2(n + 1) + cM + (1 - c)M \\ = \; & M. \end{aligned}$$

□

This proof naturally also applies to $G$. Unfortunately this property does not hold for $H$.

**Proposition 4.4.** *Let $F, G : [1, N] \to [0, M]$ continuous and non-decreasing such that for all $r \in [1, N]$ we have $F(r) + F(N + 1 - r) = G(r) + G(N + 1 - r) = M$, and let $H$ be their direct interpolation. Then in general we do* not *have $H(r) + H(N + 1 - r) = M$ for all $r \in [1, N]$.*

*Proof.* We construct a counterexample. Let $N > 2$, $p = 0.5$, set $F(x) = (x - 1)\frac{M}{N-1}$ and set

$$G(x) = \begin{cases} 0, & x < \frac{N}{2} \\ M(x - \frac{N}{2}), & \frac{N}{2} \le x \le \frac{N}{2} + 1 \\ M, & x > \frac{N}{2} + 1 \end{cases}.$$

Note that $F$ is strictly increasing and that $G$ is strictly increasing on the interval $(\frac{N}{2}, \frac{N}{2} + 1)$, such that both functions attain all values other than 0 and M exactly once. Take, say, $x = 2$, then $F(x) = \frac{M}{N-1}$. Solving $G(y) = \frac{M}{N-1}$ yields $y = \frac{N}{2} + \frac{1}{N-1}$ and this gives $H(x^{0.5} \cdot y^{0.5}) = \frac{M}{N-1}$. This implies $H$ only attains the value $\frac{M}{N-1}$ at this one point.

Next we look for $x', y'$ such that $H(x'^{0.5} \cdot y'^{0.5}) = M - \frac{M}{N-1}$. By assumption we have $F(N+1-x) = G(N+1-y) = M - \frac{M}{N-1}$ so we take $x' = N+1-x$ and $y' = N+1-y$ noting again that $H$ only attains the value $M - \frac{M}{N-1}$ at this one point. So if $H(r)+H(N+1-r) = M$ would hold for all $r \in [1, N]$ we must have

$$2^{0.5} \cdot (\frac{N}{2} + \frac{1}{N-1})^{0.5} = x^{0.5} \cdot y^{0.5} = N+1-x'^{0.5} \cdot y'^{0.5} = N+1-(N-1)^{0.5} \cdot (\frac{N}{2} + 1 + \frac{1}{N-1})^{0.5}$$

for all values of $N > 2$. It is easy to find values for $N$ such that the above equality does not hold which proves that in general we do not have $H(r) + H(N+1-r) = M$. □

It should be noted that the failure of $H$ to attain this property can be attributed to our choice to use the geometric mean when constructing the function. When using the arithmetic mean instead $H$ would inherit the symmetric structure of $F$ and $G$; in comparison the geometric mean seems to behave in a less symmetric manner. Since we want to use the geometric mean as explained last chapter we have to find a way to enforce this symmetry on $H$.

**Definition 4.5.** Let $F, G : [1, N] \to [0, M]$ continuous and non-decreasing and let $H$ be their direct interpolation. The *symmetric interpolation* of $F$ and $G$ is the function

$$\begin{cases} H(x), & x \leq \frac{N+1}{2} \\ M - H(N+1-x), & x > \frac{N+1}{2} \end{cases}.$$

Now instead of the direct interpolation we can take the symmetric interpolation as our new function $H$. Then we do have $H(r) + H(N+1-r) = M$ for all $r \in [1, N]$ and we can show that $\hat{h}$ is well-defined; from this property we have $|\hat{h}[n]| = |\hat{h}[N+2-n]|$. The phase information is transferred similarly through $H_{phase}$ enforcing that the phase contained in $H(r)$ is the opposite of that in $H(N+2-r)$. This then yields $\hat{h}[n] = \overline{\hat{h}[N+2-n]}$ or rather $\hat{h}(n) = \overline{\hat{h}(N-n)}$. When computing $h$ we find

$$\begin{aligned} h(n) &= \sum_{k=1}^{N} \hat{h}(k) e^{2\pi i n k/N} \\ &= \hat{h}(N) + \sum_{k=1}^{\frac{N-1}{2}} \Big( \hat{h}(k) e^{2\pi i n k/N} + \hat{h}(N-k) e^{2\pi i n (N-k)/N} \Big) \\ &= \hat{h}(0) + \sum_{k=1}^{\frac{N-1}{2}} \Big( \big(\hat{h}(k) + \hat{h}(N-k)\big) \cos(2\pi n k/N) + i \big(\hat{h}(k) - \hat{h}(N-k)\big) \sin(2\pi n k/N) \Big) \end{aligned}$$

so it is clear that $h(n)$ will be a real-valued function. With this last alteration of $H$ we have completed our morph and proven it is completely well-defined.

## 4.3 Algorithm

In this section we present an algorithm for computing the morph. Given two sampled audio signals $f$ and $g$ we first have to compute their Fourier transforms and the functions $F$ and $G$ (or at least $F_2$ and $G_2$ such that values of $F$ and $G$ can be interpolated when needed). The challenging part of the algorithm is computing $H$ in an efficient manner. When proving

theorem 4.1 we took some value $n$ and then searched for $x, y$ that averaged to $n$ and provided a value for $H$. This is not very practical as it would require traversing large parts of $F$ and $G$ for each $1 \leq n \leq N$, creating an algorithm of complexity $O(N^2)$. A better solution would be to construct inverse maps of $F$ and $G$ providing easy access to values for $x, y$ for any given function value. This strategy is more promising but has some slight problems. It can be difficult to find a nice level of detail since a Fourier transforms with clear peaks leads to functions $F, G$ with very steep slopes, although this could be solved using hash mappings. Another problem is that $F, G$ are non-decreasing but not necessarily strictly increasing. These factors led us to develop another strategy.

Given $F_2$ and $G_2$ we let an integer variable $x$ walk along $F_2$. For each value of $x$ we search for some $y$ such that $G(y-1) < F(x) \leq G(y)$, so for each $1 \leq x \leq \frac{N+1}{2}$ we find some value for $H$. When we have found such a $y$ for a particular value $x$ we also get that $G(y-1) < F(x+1)$ which simplifies searching for a new value for $y$. If $F(x+1) \leq G(y)$ we immediately find another value for $H$. Otherwise we have $F(x) \leq G(y) < F(x+1)$ which also yields a value for $H$ and we can increase $y$ afterwards. So essentially, while $x$ walks from 1 to $\frac{N+1}{2}$ asynchronously $y$ walks from 1 to $\frac{N+1}{2}$ as well and this gives exactly $N+1$ combinations of $x$ and $y$ that all give some value of $H$. This also ensures that these values are found in linear time.

Practically the algorithm consists of an outer loop in which $x$ is incremented and an inner loop in which $y$ is incremented. To store the values for $H$ we create an array of triples $Htemp$ where each triple represents values $x$, $H(x)$ and the phase at $H(x)$. Below is the pseudocode of the algorithm. Before this executes we assume that $F_2$ and $G_2$ are already calculated and that the phase information is stored in $F_{phase}$ and $G_{phase}$. Interpolation between phases is done using the functions $avgPhase$ and $avgPhase2$. The first is meant for morphing phases between the two functions while the second is meant for interpolating between known phases of one function. The algorithm for computing $H$ is as follows:

Initialize $Htemp$ to zero
$i \leftarrow 1$
$Htemp(i) \leftarrow (1, 0, 0)$
$y \leftarrow 2$
**for** $x \leftarrow 1$ to $\frac{N+1}{2}$ **do**
  **while** $F_2(x) > G_2(y)$ and $y < \frac{N+1}{2}$ **do**
    $i \leftarrow i + 1$
    **if** $G_2(y) = F_2(x-1)$ **then**
      $Htemp(i) \leftarrow ((x-1)^{1-p} \cdot y^p, G_2(y), avgPhase(F_{phase}(x-1), G_{phase}(y), p))$
    **else**
      $c \leftarrow \frac{G_2(y)-F_2(x-1)}{F_2(x)-F_2(x-1)}$
      $phase \leftarrow avgPhase(avgPhase2(F_{phase}(x-1), F_{phase}(x), c), G_{phase}(y), p)$
      $Htemp(i) \leftarrow ((x-1+z)^{1-p} \cdot y^p, G_2(y), phase)$
    **end if**
    $y \leftarrow y + 1$
  **end while**
  $i \leftarrow i + 1$
  **if** $F_2(x) = G_2(y)$ **then**
    $Htemp(i) \leftarrow (x^{1-p} \cdot y^p, G_2(y), avgPhase(F_{phase}(x), G_{phase}(y), p))$
  **else**
    $c \leftarrow \frac{F_2(x)-G_2(y-1)}{G_2(y)-G_2(y-1)}$

$phase \leftarrow avgPhase(F_{phase}(x), avgPhase2(G_{phase}(y-1), G_{phase}(y), c), p)$
$Htemp(i) \leftarrow (x^{1-p} \cdot (y-1+z)^p, F_2(x), phase)$
**end if**
**end for**

Now we have an array containing $N+1$ values for $H$. This does not necessarily contain all integer values of $H$ so we have to interpolate one last time. Including this interpolation the above algorithm has a time complexity of roughly $2N$. All that is left then is to construct $\hat{h}$ from $H$ and apply the inverse Fourier transform. Altogether obtaining a new fourier transform from the transformed inputs is done in linear time. Since the transforms themselves are of complexity $O(n \log_2 n)$ they have the biggest impact on the speed of the morph, which means the morph is nicely computable.

# 5 Results

Now that we have described both our morph and a way to compute it we can study the results. Since these results are audio files the reader is strongly recommended to listen to the files discussed in this chapter[12]. We will study various types of input to try to analyse the workings of the morph.

## 5.1 Simple notes

To start we take the examples shown in figures 3 and 4. We use the sounds `simplein1.wav`, `simplein2.wav` and `simplein3.wav`. The first contains a note of 80Hz, the second a note of 320Hz (two octaves higher) and the third contains two notes, one of 320Hz and one of 480Hz(a perfect fifth above). Our choice for $F_1$ or $avgPhase$ does not matter much in these cases. The resulting $F_1$ will have one or two clear peaks and nothing else and for signals with such a low number of frequencies the phases do not seem to have much influence. The given samples were produced with $F_1 = \sum_{k=2}^{n} |\hat{f}[k]|$, setting $avgPhase$ to linear interpolation and setting $avgPhase2$ to nearest-neighbour interpolation (e.g. $avgPhase(x, y, p) = x$ when $p < 0.5$ and $avgPhase(x, y, p) = y$ when $p \geq 0.5$).

When morphing `in1` to `in2` our morph behaves like the models using partials. The two peaks are identified and shifted towards each other, slowly changing the pitch of the sound. Because of our use of the geometric mean in computing $H$ the amount of changes in pitch make some amount of musical sense. Looking at the sound for $p = 0.5$ we find a frequency of 160Hz which means the note is exactly an octave above the lower note and an octave under the higher note: musically exactly in between the two. Likewise for other values of $p$ the resulting pitch lies at a frequency that makes musical sense and this helps the morphing to feel more smooth.

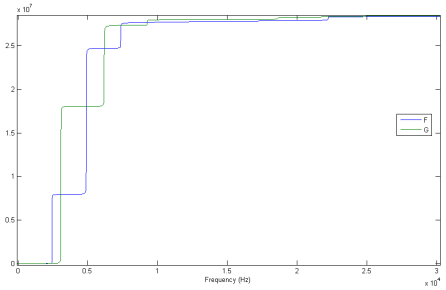
The morph of `in1` to `in3` is far less succesful. The first contains a single note and the second contains two notes that are a fifth apart, meaning their frequencies have a ratio of $2:3$. The inputs are not very similar making it less apparent what a good morphing strategy would be. Our morph behaves as in figure 4, the peak splits in two peaks that slowly shift apart. The resulting sounds are very inharmonic and have decidedly less of a smooth feel to them. During the morphing the frequencies of the two peaks go to a variety of ratios from $1:1$ to $2:3$. However almost all of these ratios give inharmonic and 'ugly' sounds that are often best avoided. These ratios are also not at all present in the input sounds which makes

it somewhat unnatural for them to show up.

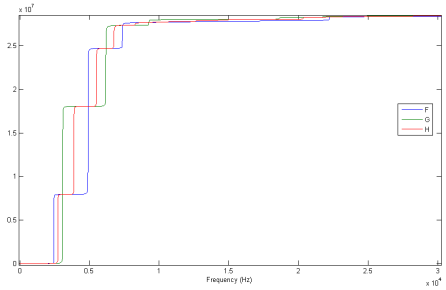
## 5.2 Musical notes

Here we study two cases. First we take two notes of the same instrument, so two sounds with a different pitch but a very similar pattern of overtones. Second we take the same note on different instruments, so two sounds with the same pitch but a different pattern of overtones. Both definitions of $F_1$ were tested, one were we sum over absolute values and one were we sum over squares. The results for the first can be found in the folder `F1 abs` and results for the second in the folder `F1 squared`.

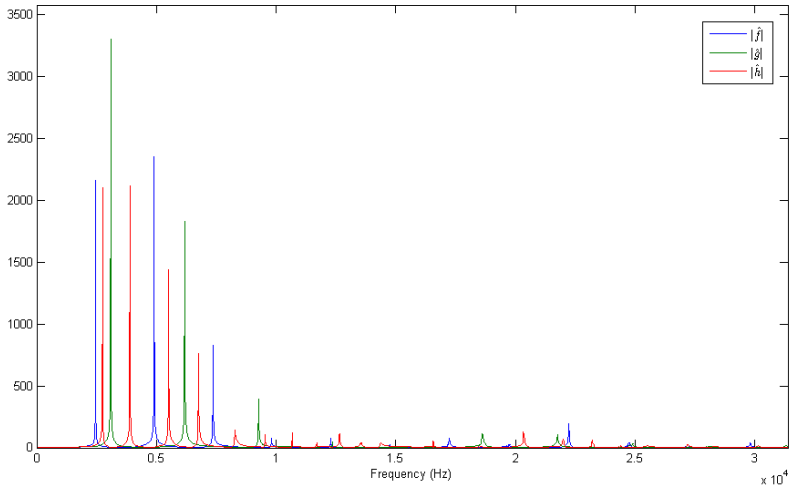
First we look at the two guitar sounds `guitar_g.wav` and `guitar_b.wav`. These sounds have a very similar pattern of overtones and all peaks in the spectrum should be clearly identifyable. One would expect the morph to produce guitar sounds slowly changing in pitch. Contrary to this the morph produces a number of samples that do not sound much like a guitar. It starts off as a guitar sound only to add inharmonious pitches and the sound quickly becomes hollow. The reason for this is that the peaks in both spectra do not at all have the same amplitudes which causes the morph to match them incorrectly. This phenomenon



(a) $F$ and $G$



(b) $F, G$ and $H$ for $p = 0.5$



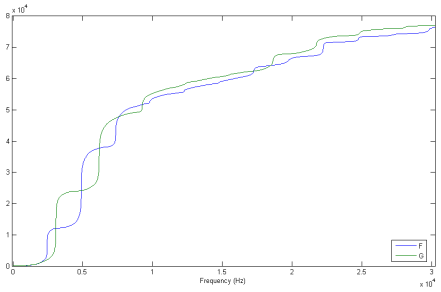

(c) Amplitude spectra for $\hat{f}$, $\hat{g}$ and $\hat{h}$ for $p = 0.5$.

Figure 5: Every time $F$ and $G$ cross creates a new incline for $H$ which in turn produces an unwanted peak in the spectrum of $\hat{h}$.

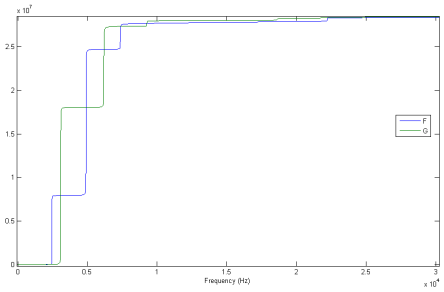
is shown in figure 5. Every time $F$ and $G$ meet prompts $H$ to create a new incline. In the

spectrum of $\hat{h}$ these lead to unwanted frequencies. For $p = 0$ or $p = 1$ these frequencies do not show up at all; for $p = 0.5$ they are most prominent as they have then distanced themselves the most from frequencies normally present in a guitar sound. This is clearly audible when listening to the morphed sounds. As the morphing progresses the unwanted sounds become more apparent and after the halfway point fade away again.

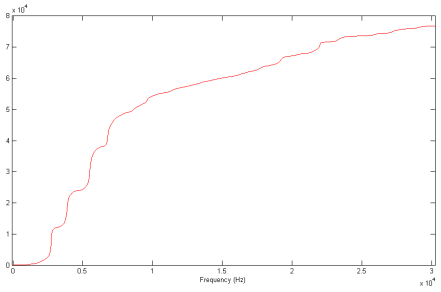
The choice of $F_1$ does have an impact on this. Using squared values instead of absolute values increases the difference between frequencies, making peaks more prominent. This can



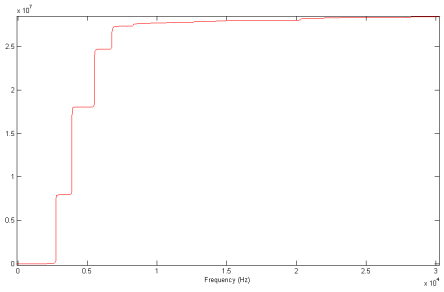
(a) $F$ and $G$ using absolute values for $F_1$



(b) $F$ and $G$ using squared values for $F_1$



(c) $H$ using absolute values for $F_1$



(d) $H$ using squared values for $F_1$

Figure 6: The choice of $F_1$ greatly impacts the shape of $F, G$ and $H$. Using absolute values accentuates the peaks less leading to less prominent unwanted frequencies.

be seen clearly in the shapes of $F$ and $G$, seen in figure 6. Since the peaks are less prominent the unwanted frequencies are less prominent as well.

Lastly it should be noted that the produced sounds have some kind of symmetrical behaviour. Like the input sounds they all start with a clear attack and gradually decrease in volume, but unlike the input sounds the volume starts increasing near the end. This is a consequence of the phase information. A Fourier transform of a signal of length $T$ decomposes that signal in sine waves that are all periodic with length $T$. The inverse of the transform can thus be seen as a $T$-periodic function as well. It is difficult to create a signal that has such a large increase in loudness at precisely one point (even though such signals occur so often in nature and thus sound more natural) and has largely to do with setting phases correctly. We tried various settings and combinations for $avgPhase$ and $avgPhase2$ which did not alleviate the problem. In the end all morphs were performed using linear interpolation and nearest-neighbour interpolation as described in the previous set.

Next we look at two different instruments playing the same note. We use the same guitar sample `guitar_g.wav` and combine it with a keyboard sample `keys_g.wav`. The exact same problems apply here. While morphing the overtones new peaks are created that are not present in either input sound and these peaks create an inharmonic sound that does not really resemble

any of the input sounds. The phase problem is a little less pronounced here; while the guitar sound has a clear attack the keyboard sound is nearly constant for its entire duration and as such the morph moves towards more constant sounds as well.

## 5.3 Noisy sounds

The previous cases considered musical sounds with a clear harmonic structure: sounds with a clear pitch and distinct or no overtones containing little noise. In these cases our morphing strategy produced unwanted overtones that did not fit the harmonic structure. It seems that such cases are better suited for clearly identifying and aligning partials. One of the areas that models using partials often struggle with is noisy sounds since there are often no clear partials and these models are usually geared towards preserving harmonic structure which these sounds do not contain. For this reason it is interesting to study the effect of our morph here as well. We use four drum sounds; two toms `tom1.wav` and `tom2.wav` and two cymbals `crash.wav` and `ride.wav`.

Looking at the morph between the tom sounds the choice of $F_1$ once again comes up. Using absolute values leads to a slightly lighter sound where the attack of the second tom becomes steadily more apparent. Squared values give a deeper sound but also a very audible high pitched noise. It also seems like the attack of the second tom has some more difficulty finding its pitch, and in general it sounds a little more disjointed.

The morph between the crash and the ride is probably the best result of this morph. The spectrum of the crash contains more frequencies and more energy than the spectrum of the ride which has an almost humble sound. The ride seems to die out a little faster as well. Overall the morph performs well here, aside from the phase issue which prevents the notes from dying out: the sounds produced by the morph seem to transition smoothly and naturally.

The previous two examples morphed sounds that were in a way very similar, often giving the listener some ideas of what an 'in-between' sound would be like. This is why the last morph we look at is between a tom and the ride cymbal which are two very different sounds. When morphing the low rumbling of the tom quickly disappears as the pitch rises. At the sample for $p = 0.7$ the distinct metal ringing of the cymbal starts to become apparent and the sound slowly solidifies from there to form the cymbal sound. It seems like the features of one sound gradually fade out while the features of the other fade in. The result is somewhat promising but we would rather have a process that feels slightly smoother.

# 6 Conclusions and further research

In this paper we presented an idea for a morph based on difficulties encountered in previous work based on partials and we presented an algorithm for computing it. The morph was supposed to avoid issues with identification and alignment of partials. There are numerous problems with this approach. At the cost of avoiding alignment of partials unwanted noise was introduced and when one partial is morphed to multiple others very inharmonic sounds appear. Lastly we have not found a nice way of handling or averaging the phase information. This has a very pronounced effect on the shape and timing of the sounds that the morph produces. In conclusion this model does not seem fit for sounds with any clear harmonic structure, as that structure is better preserved by previous models.

The results on noisy sounds without clear harmonic structure do seem interesting and it could be valuable to explore this further. In this paper we have focused mainly on musical

sounds and the peaks in their amplitude spectra. To study noisy sounds one could look at different features that are not based on partials since they seem unwieldy to use in an inharmonic context. Another point that could be explored is different strategies for handling the phase information.

# 7 References


[1] M. Payne, $L^2$ *and Fourier Series*, University of Reading, November 21, 2008.

[2] A. Freire, *Convergence of Fourier series*, Department of Mathematics, University of Tennessee. Retrieved from `http://www.math.utk.edu/~freire/m435f07/m435f07fourierconvergence.pdf` (accessed on 10-08-2015).

[3] A.V. Oppenheim, R.W. Schafer and J.R. Buck, *Discrete-time signal processing*. Upper Saddle River, N.J.: Prentice Hall. ISBN 0-13-754920-2.

[4] J. W. Cooley and J.W. Tukey, (1965). "An algorithm for the machine calculation of complex Fourier series". Mathematics of Computation 19 (90): 297301. doi:10.1090/S0025-5718-1965-0178586-1. ISSN 0025-5718.

[5] F. O'Reilly Regueiro, *Evaluation of Interpolation Strategies for the Morphing of Musical Sound Objects*, Masters thesis, Music Technology Area, Department of Music Research, McGill University, September 2010.

[6] X. Serra, *Sound hybridization techniques based on a deterministic plus stochastic decomposition model*, in Proceedings of the International Computer Music Conference(ICMC), 1994.

[7] K. Fitz, L. Haken, S. Lefvert, and M. ODonnell, *Sound morphing using Loris and the reassigned bandwidth-enhanced additive sound model: Practice and applications*, in Proceedings of the International Computer Music Conference (ICMC), 2002.

[8] T. Hikichi and N. Osaka, *Sound timbre interpolation based on physical modeling*, Acoustical Science and Technology, vol. 22, no. 2, p. 101111, 2001.

[9] M. Caetano and X. Rodet, *Evolutionary spectral envelope morphing by spectral shape descriptors*, in Proceedings of the International Computer Music Conference (ICMC), pp. 171 174, 2009.

[10] MATLAB version 8.3.0.532 (R2014a). Natick, Massachusetts: The MathWorks Inc., 2014.

[11] W. Miller, *Lecture notes on Fourier series*, supplemental material from the course Applied Fourier Analysis, University of Minnesota, 2010.

[12] `https://drive.google.com/folderview?id=0Bws15cW8Y8DlfkFjSTBpXy1mTFJQdGsxaDVUWHpWY0ZtREV4d0VRX0h6MDktVDZyTzZfLW8`