TRANSFORMATION OF AUDIO SIGNALS BY USE OF THE MCAULAY-QUATIERI SINUSOIDAL MODEL OF SOUND

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of
Master of Arts

by
Theodore Apel

DARTMOUTH COLLEGE
Hanover, New Hampshire

June 1993

Examining Committee:

(chairperson) Larry Polansky

Jon Appleton

Dean of Graduate Studies

Lee Ray

Abstract

The McAulay-Quatieri sinusoidal representation of sound extracts frequency spectra as amplitude peaks thereby providing a flexible and perceptually intuitive format for frequency domain transformations. This thesis presents some techniques techniques for sound transformation based on the McAulay-Quatieri representation. A new application for Macintosh computers developed by the author, *McAulay-Quatieri Transformer*, will be presented. It performs these spectral transformations on data created by the *Lemur* Macintosh analysis/synthesis program.

Preface

The author wishes to acknowledge:

Kelly Fitz and Bill Walker, whose Lemur program inspired and made possible this work; Professor Larry Polansky who provided unending encouragement, advice, and ideas; Dr. Lee Ray who graciously helped and encouraged this work; Mike Casey, Ko Umezaki, John Puterbaugh, and Steve Berkeley, whose coding help was invaluable; Professor Jon Appleton, who allowed this work to happen; and the other students and faculty of the Bregman Electro-Acoustic Music studio at Dartmouth College.

Table of Contents

List of Figures

Section One

Introduction

This thesis presents a sound transformation tool for use in the production of electro-acoustic music. The tool is based on the sound analysis/synthesis method of McAulay and Quatieri (McAulay and Quatieri 1986a). It will be argued that the McAulay-Quatieri (MQ) system is well suited to the task of sound transformations because it makes available important perceptual properties for manipulation. This thesis presents a specific set of sound transformation tools which demonstrate the flexibility of the McAulay-Quatieri system.

The McAulay-Quatieri analysis/synthesis system is based on modeling only those spectral characteristics of a sound which are necessary to synthesize a new sound perceptually equivalent to the original sound. Any aspects of the original sound which would be masked when the sound is heard are not encoded in the MQ system. There are two significant consequences for the system's ability to transform sounds. First, the representation uses sound parameters which correspond to perceptually significant dimensions. Second, the modification of these sound parameters results in few unexpected sonic artifacts. These features make the MQ representation a powerful technique for sound transformation. The *McAulay-Quatieri Transformer* (MQT) program presented in this thesis provides a tool for sonic transformation using the MQ representation.

The text of this thesis will present (i) relevant historical information on additive synthesis based analysis/synthesis systems, (ii) a review of the McAulay-Quatieri analysis/synthesis system, and its implementation in the Lemur program (Fitz 1992), and (iii) an overview of the MQT program and a presentation of the various sound transformation tools in MQT. A manual for using the MQT program is also included.

Section Two

Background

The modification of recorded sound for use in electro-acoustic music began with the manipulation of recorded sound on various analog physical media (Appleton and Perera 1975). This technique known as *musique concréte* involved the manipulation of sound by reversing playback, superemposing playback, slowing down or speeding up playback and looping sections of tape.

The digital computer has added greatly to the ways sounds can be manipulated. Indeed, manipulation of recorded sound by computer has almost completely replaced manipulation of any physical mediam.

The computer has also created a new range of sound transformation possibilities because the computer allows sound to be represented in new ways. Specifically, the computer allows sound to be analyzed into various representations which can be altered in various ways and then resynthesized.

Analyzing a recording of a sound in some way and synthesizing a new sound from the resulting analysis is called analysis/synthesis (Cann 1979). Several analysis/synthesis systems for transformation of digital sound are currently in common use. These systems can be grouped into subtractive-based and additive-based methods. Subtractive based methods such as Linear Predictive Coding (LPC) and Formant Wave Function Synthesis (FOF), analyze a sound for formant regions which are modeled as a time varying filter and an impulse or other excitation function. Essentially, these systems filter a complex sound with a time varying filter. Subtractive-based systems have proven to be very successful for transformation of sound in electro-acoustic music (Lansky 1989).

The second group, which we will focus on here, uses additive synthesis methods. Additive synthesis refers to the production of complex waveforms by adding sinusoids of different frequency, amplitude, and phase.

Additive synthesis has had a rich history in electronic music. Stockhausen's 1954 work *Studie II* was composed entirely by superimposing sinusoids on magnetic tape to produce complex tones. It has, however, been impractical for creating complex waveforms because of the large number of oscillators required. For this reason, frequency modulation (FM) synthesis, waveshaping and other synthesis methods have been favored because they offer complex waveforms with a minimum of oscillators (Chowning 1986). The predictability of output that additive synthesis offers has nevertheless made it popular for computer-based analysis/synthesis systems in spite of the shear computational power it requires.

The primary difficulty with analysis systems intended to drive additive synthesis is theabstracting of the sinusoidal components present in the original sound which are to be used for resynthesis. The first computer systems of this type relied on a heterodyne filter to extract the sinusoidal components (Freedman 1968, Beauchamp 1969). This heterodyne filter could only model frequency components which were harmonics of a known fixed fundamental. This meant that inharmonic sounds or those which changed in pitch were unsuitable for analysis by these systems. However, significant work in the nature of timbre perception has been carried out with such a system (Grey and Gordon 1978).

It was not until the phase vocoder technique was adapted from speech research to computer music by Andy Morrer and later Mark Dolson, that additive based analysis/synthesis became easy to use in electro-acoustic music (Morrer 1978, Dolson 1983, Dolson 1984, Dolson 1986). The phase vocoder generates additive synthesis parameters by band-pass filtering a signal into frequency bins which are at sufficiently narrow to track individual components. The output of each filter can be converted to frequency and amplitude values for synthesis. Although the phase vocoder is successful in analyzing harmonic sounds, because of its evenly spaced filter bank it is relatively unsuited to analyzing time-varying inharmonic sounds (Gordon and Strawn 1985).

More recently, the McAulay Quatieri analysis/synthesis technique has been devised to generate additive synthesis parameters of both harmonic and inharmonic sounds. It is this technique which is the basis for the transformations presented in this paper. This method produces synthesis parameters form a wide variety of sounds, making it an ideal system for these transformations.

Section Three The McAulay Quatieri Analysis/Synthesis Technique

The McAulay Quatieri analysis/synthesis technique produces an output sound which is perceptually equivalent to its input sound, while retaining only the harmonic peak information from its spectra. This Section outlines this process and introduces the Lemur implementation of this method.

The MQ representation has several important features which make it useful and interesting to the electro-acoustic composer. Before looking at the method itself, we will first look at some of the limitations of two standard representations of sound: the time domain representation and the spectral representation.

The time domain representation of sound as a series of samples is the standard method of storing sound in a digital medium. This method stores sound faithfully and reasonably compactly, but any useful sample-by-sample transformations are difficult to achieve, because of the inability to predict the sonic results of a change in the waveform. Sound transformation methods, such as waveshaping, which rely on the time domain representation of sound are hindered by this inability to accurately predict the sonic output. Nevertheless, transformations based on time domian representations are computationally efficient when compared to spectral based representations, and in many cases can yield musically useful results. Certainly transformations of larger scale time domain features such as loudness (power) are possable: gating, compressing, limiting and other dynamic processing.

Conversely, a spectral domian representation of sound has both advantages and limitations for the manipulation of sound. For example, the Short Time Fourier Transform (STFT) representation allows for modifications based on frequency spectrum.

But the inverse STFT may not produce acceptable resynthesis after modification because of the add-overlap method involved in resynthesis.

The MQ representation allows for modifications which are difficult or impossible to achieve with other representations. This results from (i) the abstraction of only perceptually salient features ignoring aspects of the sound which are masked, as well as (ii) a resynthesis method which itself allows many types of transformations.

Analysis

MQ analysis begins with performing a STFT on the analysis signal (Gordon and Strawn 1985). The first step of this transform is windowing the analysis signal in the time domain. In Lemur, the Kaiser window was chosen because it allows control of the trade off balance between frequency resolution and time resolution (Fitz 1992). Next, a Fast Fourier transform is performed on each set of windowed data to find its frequency spectra.

After a set of spectra have been computed, amplitude peaks in each spectrum are selected. This process is the defining procedure in the MQ analysis. It is assumed that sonic energy contained at these peak locations effectively masks the sonic energy at other locations in the spectrum, and that by encoding only these peaks, a perceptually equivalent set of data is preserved (Fig. 1). Perceptual experiments involving frequency masking demonstrate that this assumption is tenable for spectra with sufficiently large numbers of peaks.

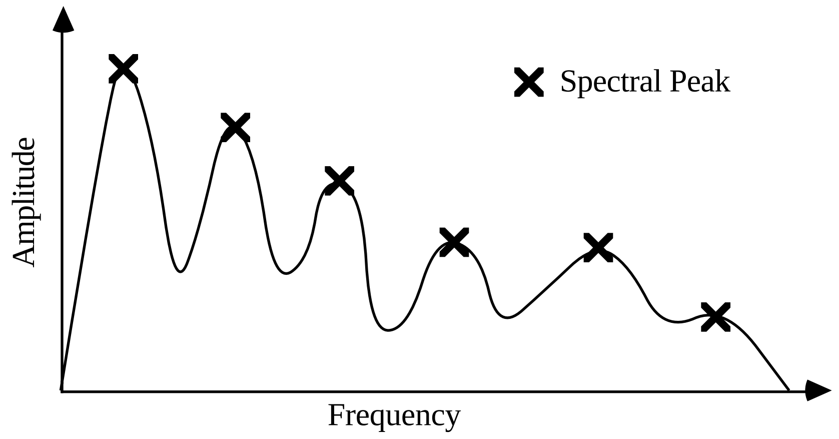


Figure 1. Spectral Peaks

The peaks are selected by finding the location of changes in spectral slopes from positive to negative. A more accurate method has been proposed by Serra in which a parabola is fitted to the peak and the location of the vertex of the parabola is encoded as the peak frequency (Serra 1989). The set of peaks abstracted from the spectrum is called a “frame”. A set of peaks in consecutive frames form a track (Fig. 2). Each frame is connected to the next using “track numbers”, which index the same peaks in consecutive frames.

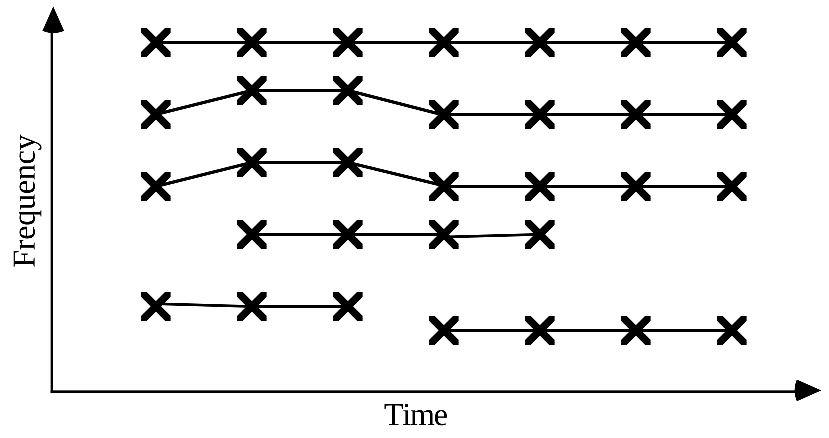


Figure 2. Track Formation

Peak-to-peak connections are formed by connecting each peak in a given frame to the peak in the subsequent frame with the closest frequency. If there is no peak within a given set frequency range, that track is not connected to the next frame; it ends. If a new peak appears that has not been connected to a previous peak, a new track begins with that peak.

After tracks have been formed, the analysis is complete. A set of frames each of which each contains a set of frequency, amplitude, and phase values along with corresponding track index number is stored as data in the MQ file (Fig. 3). This data file is the basis of the various sound modifications available in MQT.

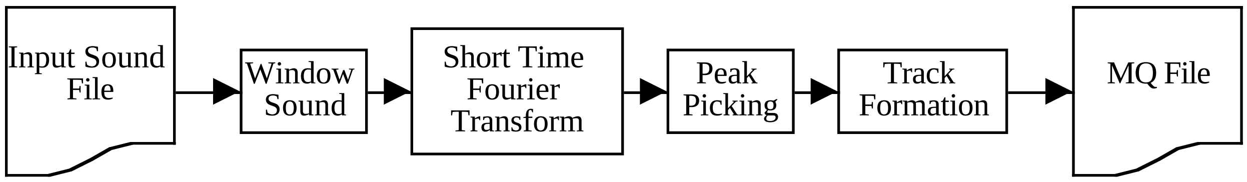


Figure 3. MQ Analysis System

Synthesis

The synthesis of a new sound from the MQ file is achieved by additive synthesis. A sinusoidal oscillator is assigned to each track and then interpolates through each subsequent peak (Fitz 1989). This method of synthesis is more flexable than the inverse STFT because of its ability to synthesize a wide range of modified sound (Fig. 4).

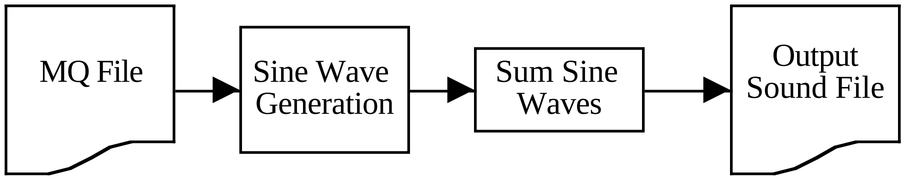


Figure 4. MQ Synthesis System

2.4 Lemur

The Lemur program by Kelly Fitz and Bill Walker is a complete Macintosh implementation of the MQ analysis/synthesis system. The Lemur program also incorporates two important sound transformation tools; pitch shifting and time compression/expansion. The amount of time compression or expansion can be varied throughout the synthesis.

The file produced by the MQ analysis in Lemur is used for all transformations in MQT. Typically, a sound is analyzed in Lemur, transformed in MQT and resynthesized into a new sound in Lemur.

Transformation of sound using Lemur and MQT is time consuming. Lemur analysis can take up to 500 times as long as the sound being analyzed. Transformation in MQT can be equally slow, and resynthesis in Lemur proceeds at approximately twice the analysis rate. Combined, the processes require a formidable amount of computing time even for very short sounds. However, special purpose signal procesing engines could perform these operations in orders of magnatude less time.

Section Four The MQT Program

The MQT program consists of several separate processing elements, each designed to transform an MQ file in some way. Section 5 through 12 discuss each of these functions in detail. Figure 5 below shows the relationship among the different elements of the MQT program. Some aspects of the separate functions in MQT remain consistent throughout the program. They are discussed in this section.

In all transformations, phase information encoded in the MQ file is left unmodified, and in some cases, may be deleted entirely. This disregard for phase information is justified by its relative unimportance when compared to frequency and amplitude information. Although leaving phase unmodified while changing corresponding frequency information could in theory introduce sonic artifacts, in practice, the sonic results are acceptable. It has been found that unmodified phase values produce results which are no worse than leaving phase information out of the analysis/synthesis entirely. However, it is assumed that better results could be achieved by modifying phase information to correspond to amplitude and frequency transformations.

Similarly, MQT makes no attempt at reformulating tracks after transformation. Track index numbers are never reassigned after a transformation in order to better match any new frequency values. As will be shown later, the ability to reassign track numbers would improve the results of some aspects of MQT.

With the exception of “track index offset”, MQT modifies frequency and amplitude values. Certainly ignoring phase and track index numbers during transformation can produce unexpected sonic results. Nevertheless, many transformations of sounds are possible with the modification of MQ frequency and amplitude values only.

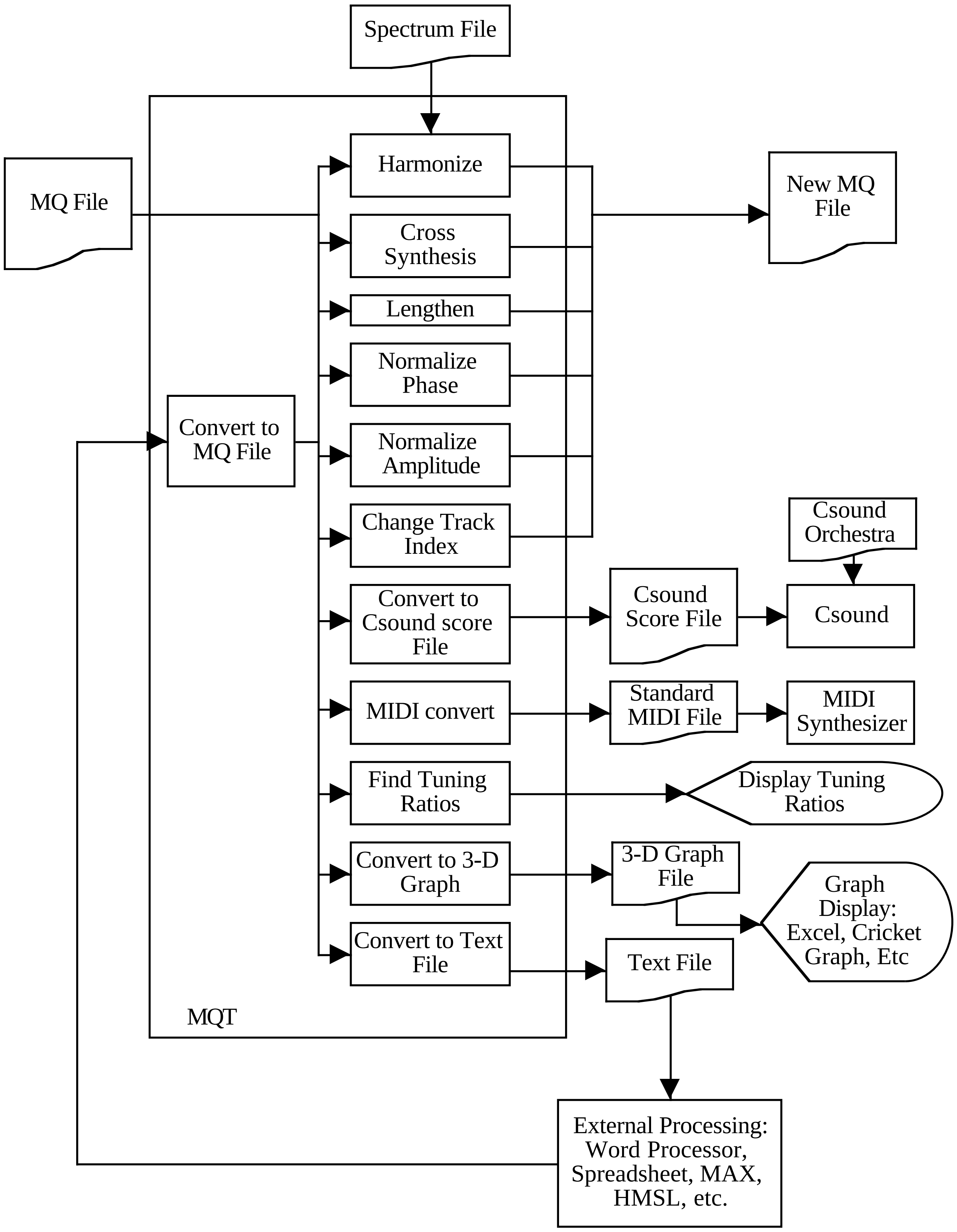


Figure 5. The MQT Program

Section Five Cross Synthesis

Cross synthesis is a strikeingly effective method of transforming sound files. Parameters from one MQ sound file are combined with parameters from another to produce a third sound file which contains elements of both. Attention must be paid to the way the files are combined for even simple changes have great effects on the resultanting hybridized sound. Before introducing these new MQ cross synthesis methods, a review of existing cross synthesis methods will be presented.

Typically, cross synthesis is carried out in a analysis/subtractive-synthesis framework. Linear Predictive coding (LPC) analysis/synthesis is often used. LPC analysis yields a sound's formant structure and excitation function. Cross synthesis may be accomplished by substituting a different excitation function before resynthesis. The result is a sound with the resonant characteristics of one sound and the harmonic characteristics of another.

Cross synthesis of additive-based analysis/synthesis data has been attempted by John Grey and John Gordon (Grey and Gordon 1978). Their method involved modeling the shape of the harmonic envelope from one sound and modifying the shape of another sounds harmonic envelope to match that shape. With this technique only the shape of the distribution of spectral energy is transformed. This type of cross-synthesis closely resembles a subtractive synthesis based approach, where the shape of the harmonic envelope can be thought of as a filter.

The cross synthesis technique possable in MQT differs significantly from these techniques. Here, a correlation between individual peaks of two MQ files is made. Neither the formant structure nor the shape of the spectra is modeled, instead individual peaks are combined to produce a new set of peaks. A new MQ file is composed of harmonic peaks which are weighted averages of the amplitude and frequency values from the two original MQ files.

MQ Cross Synthesis

Given two MQ files A and B, each frame in file A is matched to its corresponding frame from file B. If either file contains more frames than the other, any extra frames are not matched and are not included in the cross synthesis. It is therefore useful for files A and B to have approximately the same number of frames.

For each set of frames, each peak in file A is matched to corresponding peaks in file B. This is done starting at the lowest frequency in each frame and proceeding to the highest. If one frame contains more peaks than its corresponding frame, the extra peaks at the high end of the spectrum are left out of the cross synthesis. The phase and track index numbers from file A are used in the cross synthesized file.

Each amplitude and frequency value in the cross synthesized MQ file consists of some weighted average of the amplitude and frequency values from file A and B. The new amplitude values are computed by

$$A_c = ((n_a)A_a + (1 - n_a)A_b), \quad 0 \leq n_a \leq 1$$

where A_a and A_b are the amplitude values from files A and B, A_c is the new cross synthesized amplitude value, and n_a is the amplitude weighting value.

Cross synthesized frequency values are computed in a similar manner. In the frequency case, however, linear interpolations of frequency values are problematic. The average of two frequency values on a linear scale produces an unsatisfactory result. When frequencies are averaged on a base 2 logarithmic scale, the result corresponds more closely with our pitch perception. For example, averaging a 400 hz tone with a 1600 hz tone would produce a 1000 hz tone on a linear scale, and an 800 hz tone on a logarithmic scale. The 800 hz tone, perceived as an octave to the 400 hz tone, is a smoother match than a 1000 hz tone, approximately a major tenth above, would be. Therefore the frequency values are weighted on a base 2 logarithmic scale.

The frequency values are computed by

$$F_c = log_2^{-1}((n_f)log_2F_a + (1 - n_f)log_2F_b), \qquad 0 \le n_f \le 1$$

where F_a and F_b are the frequency values from files A and B, F_c is the new cross synthesized frequency value, and n_f is the frequency weighting value. Figure 6a shows a graph of an MQ representation of a singing voice sound. Figure 6b shows a similar graph of a flute sound. Figure 6c shows a graph in which the frequency values have been synthesized with 50% contributed by each sound.

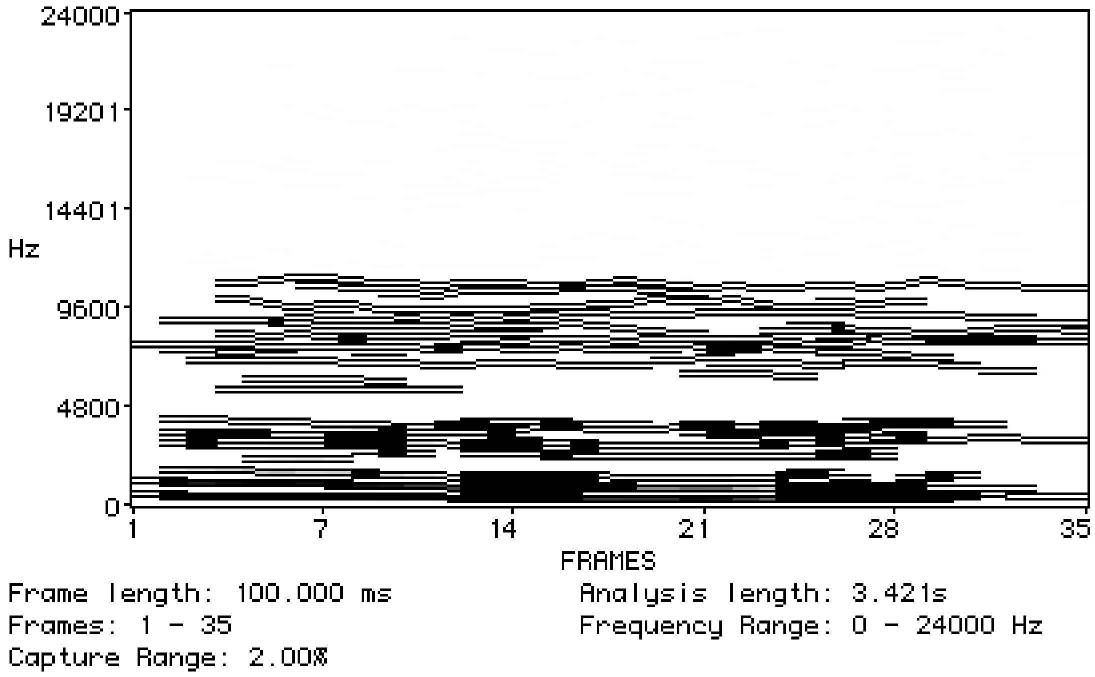
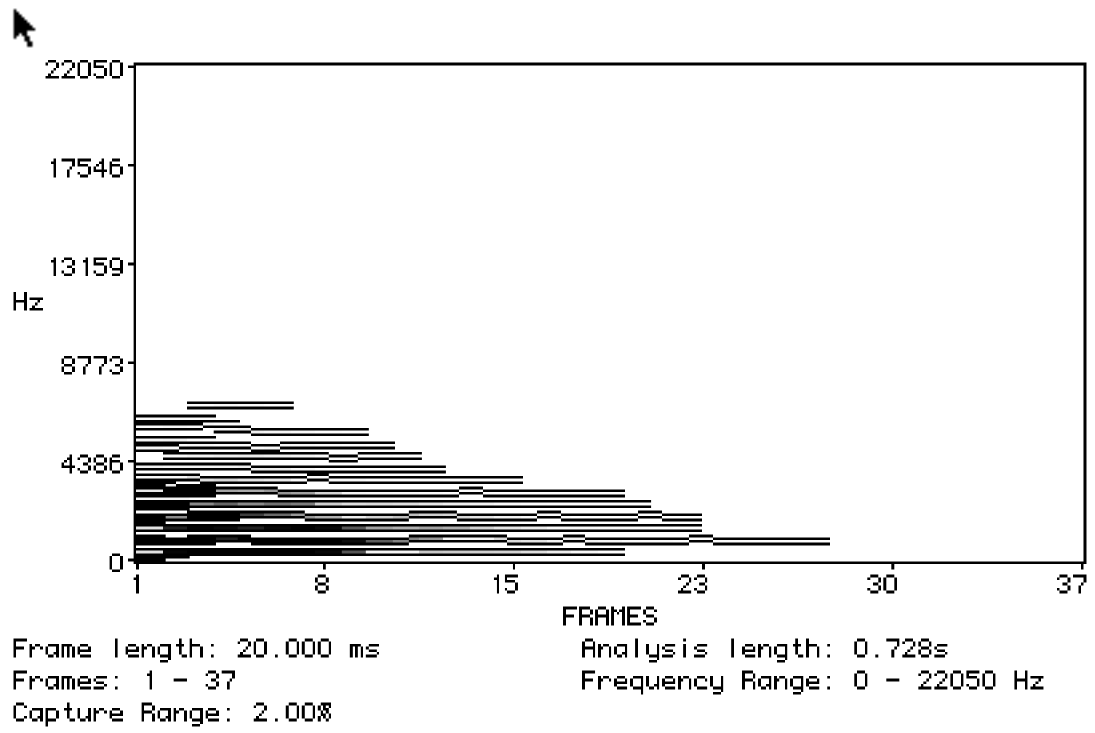
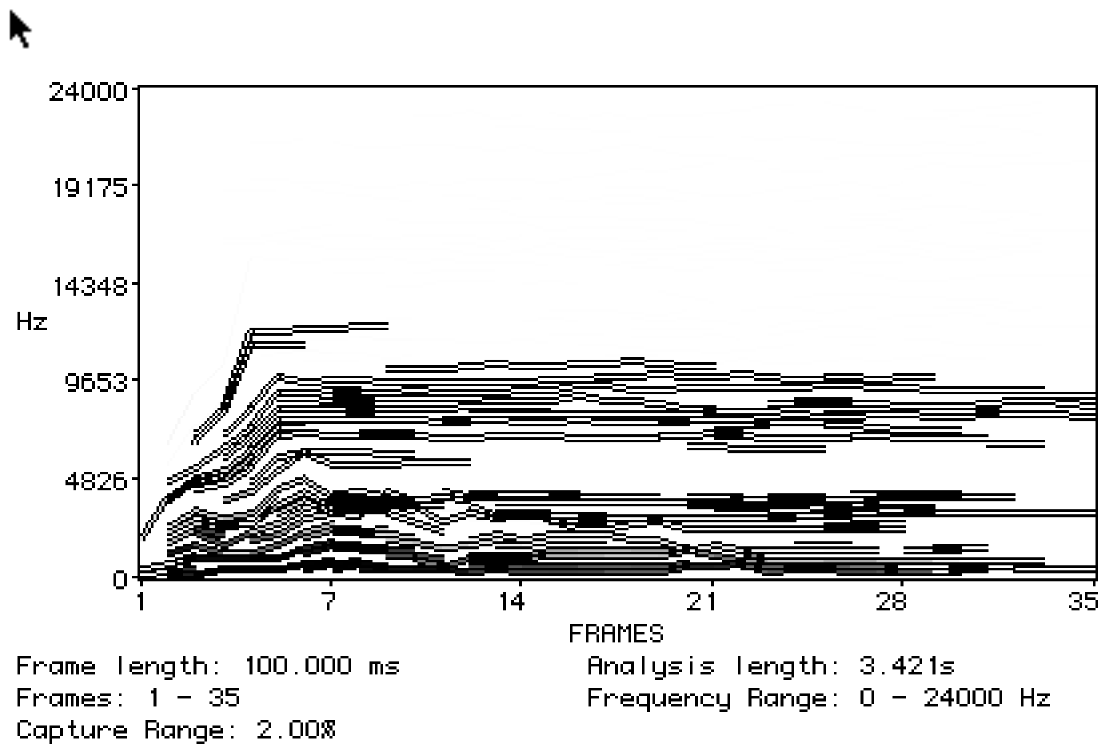


Figure 6a. MQ Analysis of Singing Voice



```
Frame length: 20.000 ms              Analysis length: 0.728s
Frames: 1 - 37                       Frequency Range: 0 - 22050 Hz
Capture Range: 2.00%
```

Figure 6b. MQ Analysis of Flute Tone



```
Frame length: 100.000 ms             Analysis length: 3.421s
Frames: 1 - 35                       Frequency Range: 0 - 24000 Hz
Capture Range: 2.00%
```

Figure 6c Frequency Cross Synthesis

The amplitude and frequency weighting value parametersn_a and n_f can be set independently. The settings for these parameters determine the relative weight of the contrabution from each file. For example, if $n_a = 1$ and $n_f = 0$ the cross synthesis will have amplitude values only from file A and frequency values only from file B.

The amplitude and frequency weighting values can remain fixed throughout the cross-synthesis or they can change. Both values can change independently from any value to any other value throughout the cross synthesis process. Figure 7 shows a graph in which the amplitude and frequency values change during the cross synthesis from 100% singing voice to 100% flute.

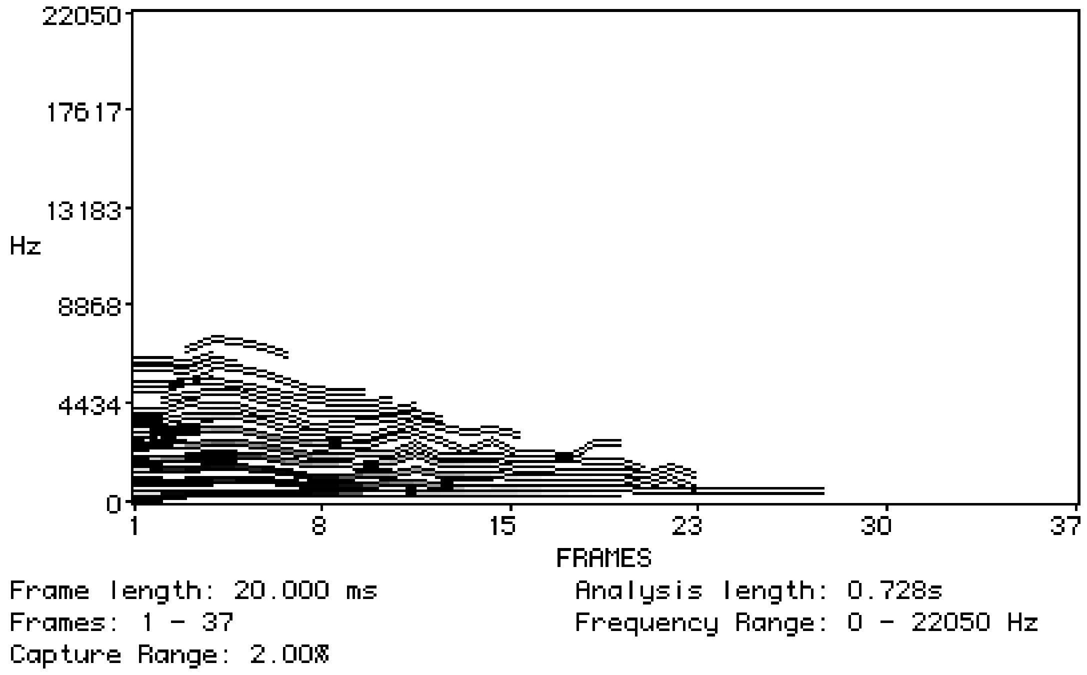


Figure 7. Time Varying Cross Synthesis

Results

The settings of the weighting parameters have a determining effect on the success of the cross synthesis. In both the static and the time varying cases, the amplitude cross synthesis is more successful in consistently producing useful hybridizations. Cross synthesis of frequency values sometimes produces interesting sonic results but seems generally less likely to yield useful results right away, at least compared to amplitude cross-synthesis. Corrently no attempt is being made to reassign track index numbers. Therefore peaks may become connected after cross synthesis which were not connected during an MQ analysis. As files are weighted farther from the file with the track numbers used for resynthesis this problem becomes worse. Misconnected tracks sound noisy and usually produce an unacceptable cross synthesis. A method for minimizing the number of these misconnected tracks is discussed below.

Time varying amplitude cross synthesis has been successful in producing new sounds. Nevertheless, some problems remain with this method. As tracks start or finish in one of the analysis files, a discontinuity is introduced in the tracking of the cross synthesized file. At worst this problem produces unacceptable results. However, a simple method can ameleviate this problem. Track amplitude hysteresis is the lowest amplitude a track may have without ending. By increasing track amplitude hysteresis during MQ analysis, the total number of tracks remains more stable than without using hysteresis (Fitz, Walker, and Haken 1992). That is, if track amplitude hysteresis is high, then tracks will not die until their amplitude reaches a very low level. In fact, there amplitude could be zero sometimes. One benifit of using high track amplitude hysteresis is stableizing of the number of tracks, thereby reducing or eliminating discontinuities in tracking to produce a smoother cross synthesis.

The linking of tracks remains a fruitful source of future work. Here are two possabilities. First, after cross synthesis, new track index numbers could be assigned using the same track picking procedure as in the MQ analysis algorithm. Each peak would be reassigned a new track number based on its similarity to a peak in the subsequent frame.

Second, the cross-synthesis method itself could be modified to correlate tracks together instead of individual peaks. Cross synthesizing would move entire tracks in frequency and amplitude. This would solve the problem of discontinuity between frames.

Nevertheless, the method employed in MQT can result in interesting new hybrids of sound and it suggests areas for future work.

Section Six

Reharmonizing

The usual time and frequency representations of sound do not easially allow changes of individual frequency components of a sampled sound. In the time domain representation of sound, individual frequency components are impossible to separate out of the composite signal. In the frequnecy domain representation, individual partials of a sound are not specifically present as such and are therefore difficult to modify. Information about the overall harmonic content is present, but in a form which does not readially lead to modifications which both extend and preserve the character of the original. Also, as previously discussed, arbitrary modification of frequency components in a SFFT representation can produce unwanted artifacts during the reverse FFT.

Analysis and subsiquent additive resynthesis permits modification of individual frequency components in a simple powerful manner. However, the shere number of sinusoidal components of complex sounds can make additive resynthesis difficult. Since MQ analysis yields only perceptually salient componenats, the power of additive synthesis can be harnessed to modify frequency components of an MQ sound file to match any predefined frequency spectrum. In such a frequency component transformation the amplitude envelope characteristics of the individual components of the original sound are preserved while the frequency characteristics are altered to match a given model.

Additive synthesis parameters have long been avalable for modification, however, such parameters are typically not derived from analysis of a recorded sound but are constructied from an abstract model. Such an approach has the advantage of allowing harmonic relations to be explicitly specified, but the resultanting sound can be lacking in the interest and detail we associate with lively sounds.

By modifying MQ data, complex amplitude envelopes can be retained from the analyzed sound while still allowing any given relationship among the partials of the sound to be specified. Thus, sounds can be constructed to conform to abstract frequency models, while retaining the complex time varying amplitude envelopes of there original forms.

In MQT, the “reharmonize” function iaccomplishes this. After reharmonizing, all frequencies are grouped at the values determined by the frequency model while amplitude values retain their original shape. Frequency models are specified as a set of frequency values produced and stored in a separate file. This file consists of a set of up to 100 frequency values which are used to transform the frequency content of the MQ file. Those frequency values in the MQ file which are closest to any of the frequencies in the frequency model file are reassigned to that new frequency value. For example, if f_1 through f_n is a set of frequency values which describe the target frequency modal, and f_{mq} is a frequency value from the MQ file, then

$$\text{if} \quad \left(\frac{f_1 + f_2}{2}\right) \leq f_{mq} \leq \left(\frac{f_2 + f_3}{2}\right)$$

$$\text{then} \quad f_{mq} = f_2$$

Figure 8 shows an MQ analysis of a Piano tone.

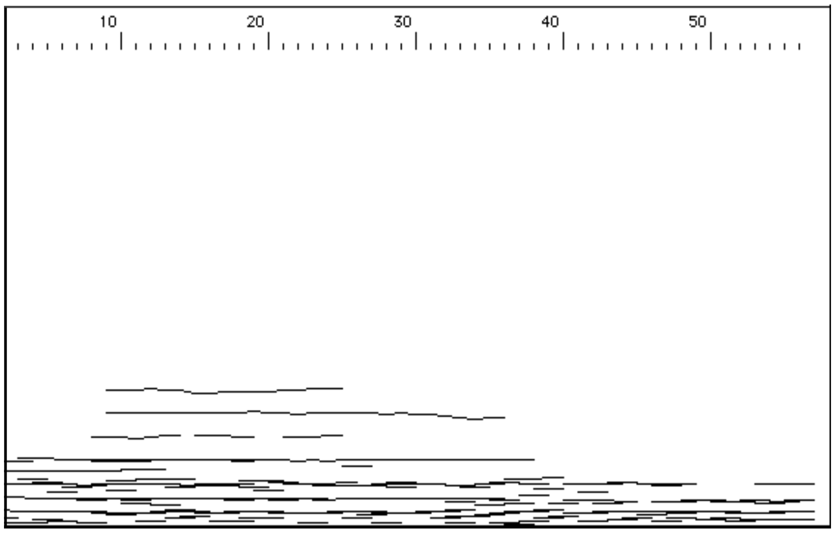


Figure 8. MQ Analysis of Piano Tone

Figure 9 shows the same tone after it has been reharmonize to a set of harmonics which are evenly spaced in frequency.

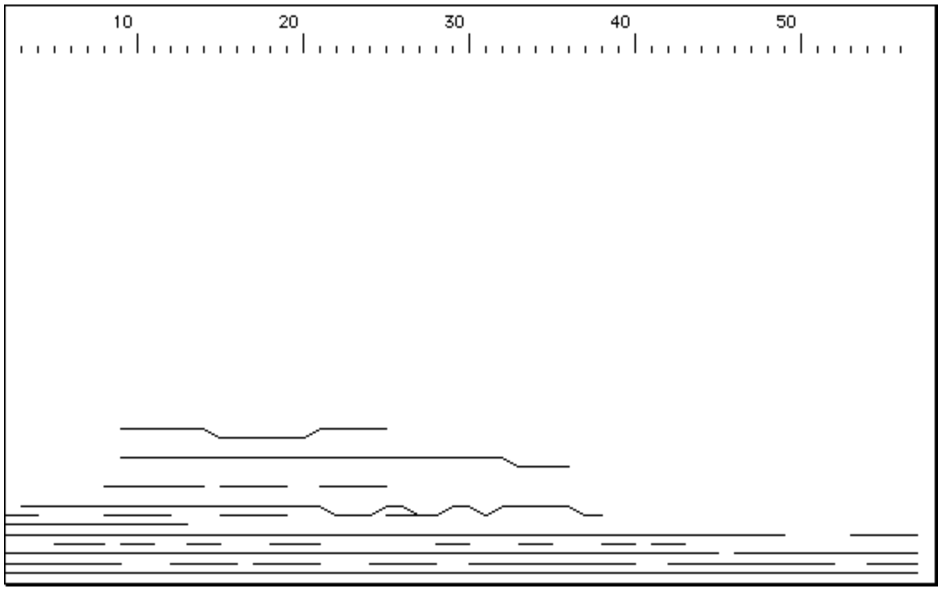


Figure 9. Reharmonized Piano Tone

Note that in Figure 9, a track will occasionally be seen to jump from one frequency to another. This occurs when a single track is being grouped at different frequencies during the life of the track. While this can add a small amount of graininess to the resulting sound, good results are nevertheless possable with this transformation.

Results

Reharmonizing allows individual frequency components of natural sounds to be systematically altered to match a target frequency model while preserving there time varying amplitude characteristics. For example, inharmonic sounds such as a gong stroke have been reharmonized to a target frequency model consisting of integer multiple harmonics. The resulting sound contains components which are harmonically related but which evolve over time in the complex way associated with the percussive source. Further, the harmonic spectrum of a piano has been modified to conform to harmonics in the Pierce scale. Such reharmonizations will be discussed further in section 9.

Section Seven

Lengthening

Changing the length of a sound without changingits pitch is an important and useful transformation and one which motovates the use of analysis/synthesis methods. Such transformation is typically accomplished by changing the rate at which resynthesis is carried out. For example, in Lemur, time scale modifications of sound are achieved by changing the length of time between synthesis of frames (Fitz 1992). For instance, time stretching is achieved by synthesizing each sine wave for a correspondingly longer duration. Lemur does this quite well. However, slowing down the rate of frame sreynthesis, there is a corresponding decrease in the detail of the sound. In some cases, such as unvoiced vocal sounds, noise is erroniously transformed into pitch successions.

MQT allows a new method of time stretching in which the duration of a sound is increased while reataining the rate at which the sound is synthesized. This allows sounds to be extended in duration while still retaining detail. MQT accomplishes time stretching by building a new MQ file with frames which are recycled back into the sound. Each frame in the new MQ file is followed by the frame that directly preceded it in the original MQ file. The resultant ing MQ file contains twice as many of frames as the original. Figure 10 shows how this is done. If the top line of numbers represents the frame numbers from the original MQ file, then the second line of numbers represents the order of frames after the lengthening transformation.

Frame Number

Figure 10. Frame Lengthening Method

Results

Although time strecthing in MQT is relatively primitive, the resultanting sound can be more interesting than those produced by changing the synthesis rate. Time stretching could be improved and expanded in a number of ways. A more complex recycling algorithm could be introduced, and a variable rate of recycling could be added. Still, this function is successful in lengthening a sound while retaining a high synthesis rate and the attendant preservation of sonic detail from the original sound.

Track index numbers in MQ files connect peaks in consecutive frames into tracks. Each track is synthesized as a sine wave in frequency and amplitude in order to pass through each peak of the track. During analysis these track numbers are selected to correctly track frequency components through time. Therefore, any transformation of these numbers will produce tracks which incorrectly track frequency components.

However, deliberately offseting track numbers can produce interesting new sound transformations. The track index change function in MQT allows the track indexes in each frame to be offset by any factor. This has the result of mismatching peaks by the given amount. For example, if the track indexes were offset by +1, each peak would be connected to the peak above it in the next frame. Figure 11 shows an MQ analysis of a Piano Tone after the track index numbers have been offset by +1.

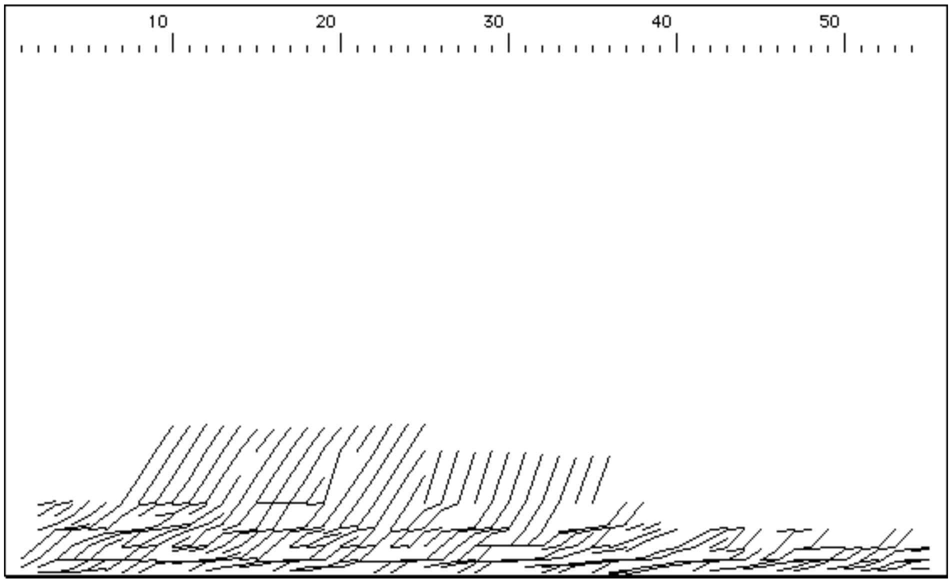


Figure 11. Piano Tone After Track Index Offset

Results

The results of this transformation have been only relatively successful in producing interesting new sounds. After any misalignment of tracks, the subsiquent resynthesis becomes quite noisy. Interesting sounds have been produced when the distance between analysis frames is very large. By increasing the distance between analysis frames, there is more time for tracks to change pitch between frames, and resynthesis is smoother. However MQ analysis with a longer frame rate produces correspondingly less detail in the sound.

Section Nine Abstracting Harmonic Ratios

The frequency values encoded in an MQ file can by used to represent harmonic relationships present in the sound. Abstracting these harmonic relationships can be useful in research and composition. When octave reduced, and expressed as ratios of harmonic to fundamental, these ratios can be interpreted as tuning ratios. This type of analysis can be important in psychological and compositional work involving the relationship between timbre and tuning.

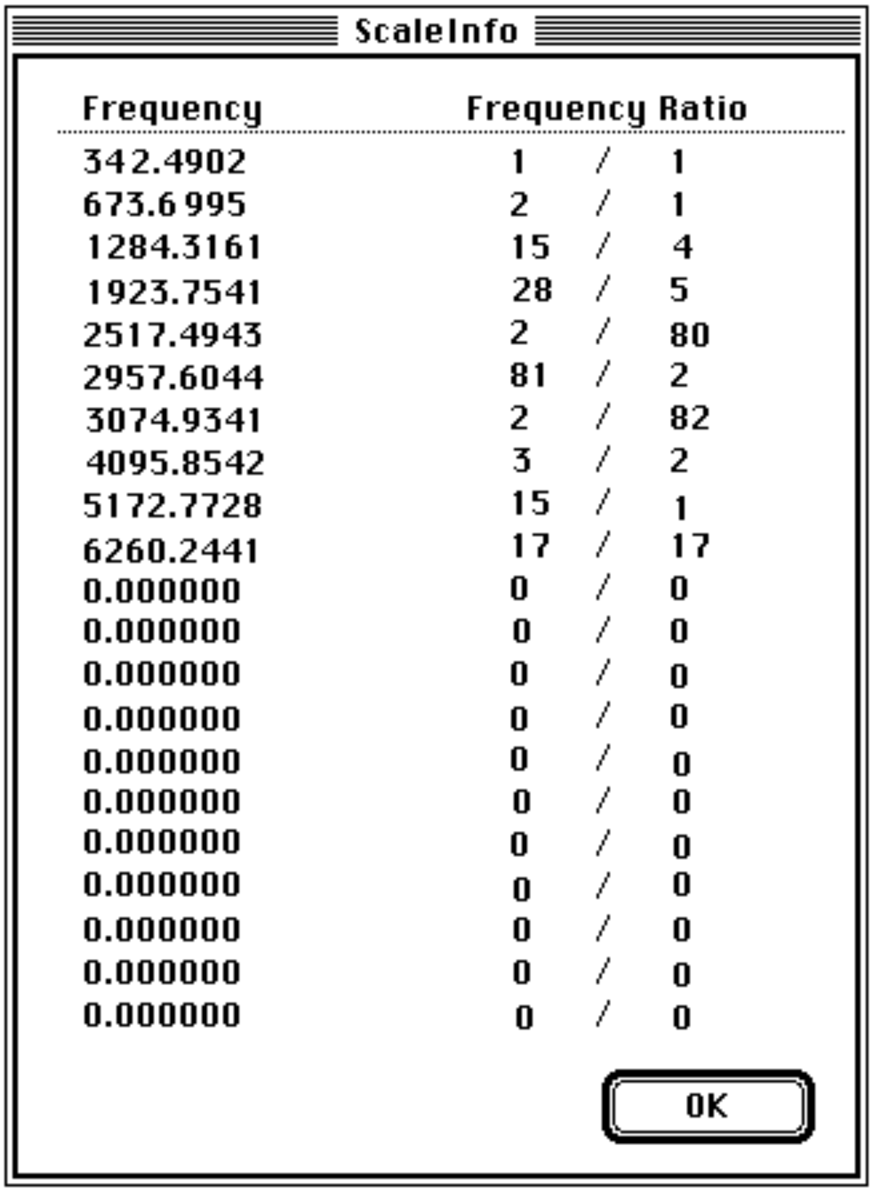
Before describing how MQT can abstract harmonic ratios from an MQ file, a very brief discussion of the relationship between timbre and tuning is in order. Research on this relationship involved stretched partials of harmonic sounds and their relationship to perceived consonance and dissonance (Slaymaker 1970). This work suggested that, intervals are consonant when no two partials fall within a critical band (Plomp and Levelt 1965). Pierce suggested that consonance could be achieved in arbitrary scales by building the partials of a sound from the ratio of the fundamental to the tones of the scale (Pierce 1966). The success of his experiments led him to conclude that “by providing music with tones that have accurately specified non harmonic partial structures, the digital computer can release music from the tyranny of 12 tones without throwing consonance overboard.” Pierce used this idea in his piece “Eight Tone Canon” in which the octave is divided into eight equal intervals and partials having octave, half octave, and quarter octave partials (Pierce and Mathews 1969).

In addition to research in building new tuning systems which can be integrated into the timbre of the sounds in which they are played, research has been done in analyzing the relationship between timbre and tuning in existing instruments and music. Young suggests that the stretched octaves of the piano derive from the inharmonicity of the piano wire sound itself (Young 1952). Carterette, Vaughn, and Jairazbhoy suggest that the Tambura drone spectrum is intrinsically tied to the Rag scales in North Indian classical music (Carterette, Vaughn, and Jairazbhoy 1989).

The MQ file provides an ideal framework for further work in these areas. The ability to abstract harmonic ratios is a first step in this direction.

Method of Analysis

In order to abstract harmonic ratios, MQT first finds one frame from the MQ file to analyze. The frame which is half way through the file is selected. Though the choice is arbitrary, this frame may be considered representative of the “steady state” of the sound. After this frame is selected, the twenty peaks with the highest amplitude are selected and ordered from highest to lowest amplitude. The lowest frequency peak in this set is assumed to be the fundamental and the other peaks are octave-reduced to the fundamental. Each frequency value is compared to the others to insure that it is unique, and ratios are made between each frequency value and the fundamental. These ratios are reduced to small integer ratios and displayed to the user. Figure 12 shows the harmonic ratios abstracted from a piano tone.



Frequency	Frequency Ratio
342.4902	1 / 1
673.6995	2 / 1
1284.3161	15 / 4
1923.7541	28 / 5
2517.4943	2 / 80
2957.6044	81 / 2
3074.9341	2 / 82
4095.8542	3 / 2
5172.7728	15 / 1
6260.2441	17 / 17
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0
0.000000	0 / 0

Figure 12. Harmonic Ratios of Piano Tone

Results

The relationship between timbre and tuning is musically significant and relatively unknown. MQT's ability to abstract some harmonic ratios from a sound should help researchers better understand the relationship between the timbre and tuning of an instrument. Further, new tambres could be constructed, even from recorded natural sound, that would match the ratios which underly the tuning of various scales.

Section Ten Converting to MIDI File

The MQ analysis file can be thought of as a set of instructions for synthesizing a sound. The MQ file format which is used by Lemur is suitable for resynthesis only by Lemur itself. But modifying the format of a Lemur MQ file, the MQ analysis may be used with other synthesis methods. By synthesizing an MQ file externally to Lemur, all the native abilities of the external synthesis scheme can be used to transform sound. This may take the form of modifying the MQ data in the context of the new platform, or synthesizing the sinusoidal components of the MQ file with other waveforms. For example instead of a sine wave, a narrow noise band could be used for resynthesis.

This section and the following section describe transformations which change an MQ file into data for use with Csound and MIDI systems (Vercoe 1990). Transformation into each of these formats requires modifying the MQ file in different ways. These modifications will be presented, as well as possible sound transformations in the new format.

Method of MIDI file conversion

The MIDI file conversion function of MQT converts an MQ file into a type 1 standard MIDI file (SMF). This allows approximations of MQ files to be synthesized on MIDI instruments. The capability to synthesize MQ files on MIDI instruments provides a tool for composers working with the relationship between timbre and note selection.

In order to convert an MQ file to a MIDI file, many simplifications and changes must be performed. The MQ file must first be reduced to only 16 tracks. This is done by converting only the 16 tracks with the highest amplitudein order to conform to the MIDI standard. Frequency values are selected by choosing the note number closest to the frequency in the MQ file. This is done using the formula

$$n = \left(\frac{ln\left(\left(\frac{f}{440\ hz}\right)^{12}\left(2^{69}\right)\right)}{ln\ 2} \right)$$

Where *n* is the MIDI note number and *f* is the frequency form the MQ file. Similarly, amplitude values in the MQ file are scaled to MIDI amplitude values between 0 and 127.

The time between each set of MIDI notes is derived from the analysis rate, so the total duration of the MIDI file is the same as the MQ file. If time scaling or other modifications to the MIDI file are desired, they can easily be accomplished in a sequencer program. Figure 13 shows a sample MIDI file which was produced by MQT. This file has been converted to text from the binary file which MQT produces.

```
Header: format 1 #tracks 1 division 480
*** Track start
0000000: Tempo: (μsec per 1/4 note) 500000
0000000: Note on:  chl 1 pitch 47 vel 55
0000002: Note off: chl 1 pitch 47 vel 55
0000002: Note on:  chl 1 pitch 48 vel 55
0000004: Note off: chl 1 pitch 48 vel 55
0000004: Note on:  chl 1 pitch 47 vel 57
0000006: Note off: chl 1 pitch 47 vel 57
0000006: Note on:  chl 1 pitch 48 vel 57
0000008: Note off: chl 1 pitch 48 vel 57
0000008: Note on:  chl 1 pitch 47 vel 57
0000010: Note off: chl 1 pitch 47 vel 57
0000010: Note on:  chl 1 pitch 48 vel 56
0000012: Note off: chl 1 pitch 48 vel 56
0000012: Note on:  chl 1 pitch 47 vel 56
0000014: Note off: chl 1 pitch 47 vel 56
0000014: Note on:  chl 1 pitch 48 vel 54
0000016: Note off: chl 1 pitch 48 vel 54
0000016: Note on:  chl 1 pitch 47 vel 55
0000018: Note off: chl 1 pitch 47 vel 55
0000018: Note on:  chl 1 pitch 48 vel 55
0000020: Note off: chl 1 pitch 48 vel 55
0000020: Note on:  chl 1 pitch 47 vel 53
0000022: Note off: chl 1 pitch 47 vel 53
0000022: Meta event, end of track
*** Track end
```

Figure 13. Sample MIDI File

Results

This transformation has unexpectedly given rich and interesting results. The author's composition "Rykoff Sexton" for soprano and tape was composed with this transformation. MIDI files derived from the harmonic content of a singing voice were synthesized on various MIDI instruments and recombined with the original singer. Thus, the process appears to have been successful in deriving pitch material which corresponds closely with the spectra of an acoustic source.

The MIDI file conversion feature of MQT may also serve as a starting point in future work in attempting to solve the elusive problem of blind score extraction from an acoustic source. An attempt in this direction was made using a recording of Erik Satie's piano work "Vexations". Figure 14 shows the score for the beginning of that composition.

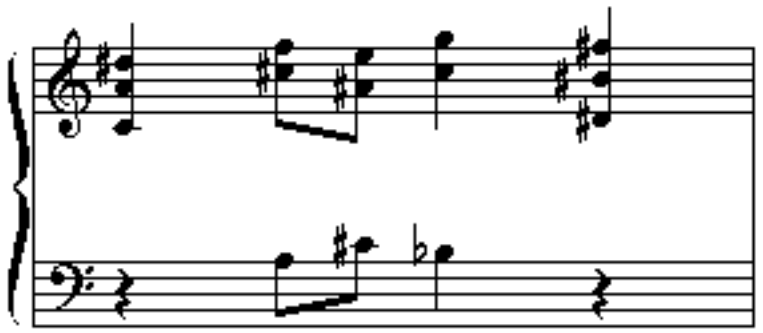


Figure 14. Beginning of score for Erik Satie's "Vexations"

A recording of this section was analyzed in Lemur with a frame rate of 150 ms and a very low peak threshold. This is an extremely long frame rate and would produce a very poor resynthesis in Lemur since frame rates are typically around 3 ms. The analysis file was converted to a MIDI file in MQT and that MIDI file was opened in the music notation program "Finale". Figure 14 shows the Finale score of the resultant MIDI file.

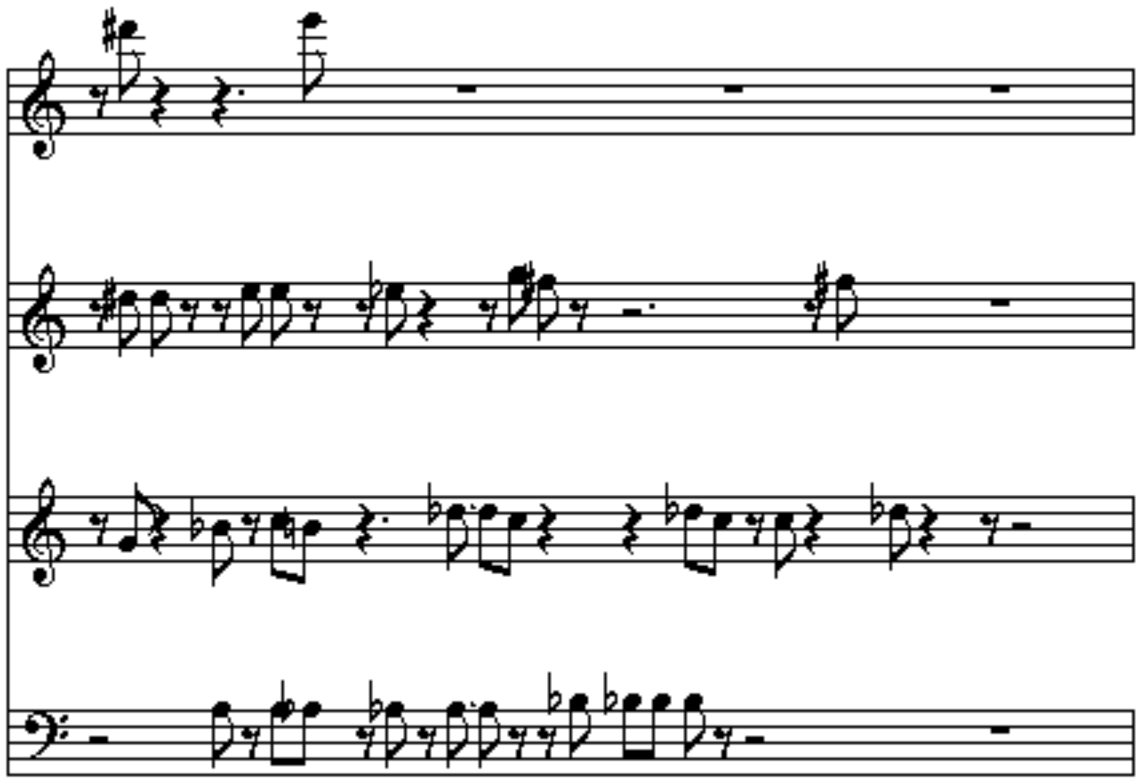


Figure 15. Score of MIDI File Derived from Recording of "Vexations"

It is immediately evident is that the process produced many of the notes in the original score, as well as notes which are one octave above the notes in the score. Although timing information is not representative to the score, these results are significant and promising because in addition to being a fruitful source of material for electro-acoustic music composition, they suggest directions for work in complete score abstraction.

Section Eleven Converting to Csound Score Files

Csound is a computer language designed to produce computer music. It is both general and powerful, allowing a variety of synthesis methods and controls for those methods. Csound uses the idea of an “orchestra” file and a “score” file. The orchestra file specifies the synthesis method to be used to produce a sound, while the score file specifies what is to be played by the orchestra. MQ data can be used to create a Csound score file. A straightforward resynthesis of the original analyzed sound can be accomplished in Csound by playing this score by a sine wave orchestra file. This process is equivalent to the resynthesis which occurs in Lemur where a sine waves essentially play the tracks of the MQ file. By resynthesizing in Csound, many new types of transformations are possible. For example, each sine wave component of the MQ file could be resynthesized with a waveform other than a sinusoid. Figure 15 shows a Csound score produced from analysis of a piano tone.

```
f1 0 2048 10 1
i1  0.000000      0.002902      1197.480      137.264328
i1  0.000000      0.002902      832.0034      387.904144
i1  0.000000      0.002902      494.1508      678.450012
i1  0.000000      0.002902      505.3681      1177.25708
i1  0.000000      0.002902      300.4258      1315.59277
i1  0.000000      0.002902      145.3233      1593.89440
i1  0.000000      0.002902      41.31154      1930.25695
i1  0.000000      0.002902      80.62171      2133.81005
i1  0.000000      0.002902      116.4693      2672.53002
i1  0.000000      0.002902      23.14518      2938.54052
i1  0.002902      0.002902      1136.489      126.946320
i1  0.002902      0.002902      788.9614      397.203735
i1  0.002902      0.002902      427.8754      626.445496
i1  0.002902      0.002902      169.0656      820.591797
i1  0.002902      0.002902      118.5320      931.798645
i1  0.002902      0.002902      493.9532      1176.65014
i1  0.002902      0.002902      188.7016      1544.99987
i1  0.002902      0.002902      20.35580      1789.05407
i1  0.002902      0.002902      95.88626      2099.56152
i1  0.005805      0.002902      1157.979      131.064865
i1  0.005805      0.002902      735.3663      388.792145
i1  0.005805      0.002902      435.4026      684.729187
i1  0.005805      0.002902      606.4624      1206.24218
i1  0.005805      0.002902      152.6111      1594.88537
i1  0.005805      0.002902      40.29315      1933.45654
e
```

Figure 16. Csound Score file

Results

The Csound orchistral score paradime has produced succesful resynthesis transformations of MQ files, first by adding harmonics to each sine wave comonent and second, by adding vibrato to all the components. Because track index numbers are not translated to the Csound score, Csound is only able to make an approximation of the MQ file. For example, if a track in an MQ file increases from 440 hz. to 402 hz. between frames, the Csound score would play a 400 hz tone for the duration of the frame and then play a 402 hz tone for the duration of the following frame. This is equivalent to sampling the MQ file at the frame rate of the MQ analysis. The problem can be reduced by setting the frequency drift to a narrow range during analysis. The data produced from such settings would minimize the discontinuities at each frame.

Section Twelve Converting To and From Text files

The final type of MQ transformation possible to be described is the simplest, but potentially the most powerful. By simply converting the binary information of a Lemur file to text, a user is free to transform the MQ file in a detailed, specific, and arbitrary manner. The instantion of analysis data as text makes many methods of transformation are possible. For example, the data could be imported to a spreadsheet where changes could be made by row or column; the data could be transformed in a general computer music language such as HMSL or MAX; and ultimately individual data elements of an MQ file could be modified by hand in a text processor.

Text files generated by MQT can be converted back into MQ files after modification. It is even possible to use this ability to synthesize completely new sounds by generating as text all aspects of an MQ file. Figure 16 shows the format of an MQ text file produced by MQT. This format follows the format of a Lemur File.

```
2175
104
-80.000000
20.000000
20.000000
378.557404
40.000000
0.100000
164224
48000.000000
0.020000

1

0.219370   1221.256470   -2.292239     1

1

0.229670   1210.045410   1.625257      2

2

0.000000   1181.279785   -0.980449     2
0.326698   1208.188477   1.023958      3
```

Figure 17. Beginning of MQ Text File

The file begins with the header information. (The meaning of this information can be found in the Lemur Manual) All theframes of the analysis file come next. Before each frame is the number of spectral peaks in that frame. Each frame consists of the linear scale amplitude (between 0 and 1), frequency, phase, and track index number all separated by tabs.

Results

While MQ text files can be lengthy and the manual modification of their parameters can be laborious, the conversion to and from text files provides an additional tool for researchers and composers to make specific transformations to an MQ file.

Section Thirteen

Conclusions

We have seen that the McAulay-Quatieri representation of sound provides a powerful format for transformation of sounds. By encoding spectral peaks as separate sinusoidal components, many novel types of modifications heretofore little used in electro-acoustic music are possible. Several such transformations which take advantage of the MQ representation have been presented. The MQT program, introduced here, has been successful in carrying out these transformations on a Macintosh computer.

Certainly areas in need of refinement remain. The transformation of peak information in MQT without regard to its function as part of a track introduces some difficulties. Future work with MQT should preserve track identity within MQ analysis files.

Other areas of exploration could involve transformations based on the ideas of source separation as described in the work of Dan Ellis (Ellis 1992). Given an acoustic signal containing more than one simultainously sounding source, source separation techniques ould allow perceptually distinguishable sounds to be treated differently.

Second, real time transformations of sound based on the MQ representation may be possible as computer processing becomes faster. Real time synthesis of MQ files is already available in the Capy Bara system of Kurt Hebel and Carla Scaletti (Scaletti 1989) . Real time MQ type analysis/synthesis systems will certainly appear in the future. Ultimately, some of the transformations presented here may be avalable as part of a real time analysis/synthesis synthesizer.

Finally, the MQ file provides an excellent abstraction of sound as a basis for musical score extraction. While it is currently difficult for even monophonic melody lines to be abstracted blindly as a score from a sound input, the MQ representation would seem to be a promising one for beginning such work.

Appleton, Jon. and Ronald Perera. "The Development and Practice of Electronic Music." New Jersey: Prentice-Hall, 10-11.

Beauchamp, James W. 1969. "A Computer System for Time-Variant Harmonic Analysis and Synthesis of Musical Tones," in Heinz von Foerster and James W. Beauchamp, eds. *Music by Computers*, New York: Wiley, 19-62.

Bracewell, Ronald N. 1978. *The Fourier Transform and its Applications*, New York: McGraw-Hill.

Cann, R. 1979-1980. "An Analysis Synthesis Tutorial." Part 1, *Computer Music Journal* 3(3):6-11; Part 2, *Computer Music Journal* 3(4):9-13; Part 3, *Computer Music Journal* 4(1):36-42.

Carterette, Edward C., Vaughn, K., Jairazbhoy, N.A. 1989. "Perceptual, Acoustical, and Musical Aspects of the Tambura Drone," *Music Perception:* Vol. 7, No. 2, 75-108.

Chowning, John. and David Bristow. *FM Theory and Applications.* Yamaha Music Foundation. 1986.

Dodge, Charles and Thomas A. Jerse. 1985. *Computer Music*, New York: Schimer Books.

Dolson, Mark B. 1983. "Musical Applications of the Phase Vocoder," *Proceedings of the International Computer Music Conference.* 1983. Rochester, New York. 99-102.

Dolson, Mark B. 1984. "Refinements in the Phase-Vocoder-Based Modification in Music." *Proceedings of the International Computer Music Conference.* 1984. Paris, France. 65-66.

Dolson, Mark B. 1985. "Recent Advances in Musique Concrete at CARL." *Proceedings of the International Computer Music Conference.* 1985. Burnaby, B.C., Canada. 55-60.

Dolson, Mark B. 1986. "The Phase Vocoder: A Tutorial." *Computer Music Journal*. 10(4): 14-27.

Dolson, Mark B. 1989. "Fourier-Transform-Based Timbral Manipulations." *Current Directions in Computer Music Research.* Max Mathews and John Pierce ed. Cambridge, Massachusetts: The MIT Press: 105-112.

Ellis, Daniel P. 1992 "A Perceptual Representation of Audio." M.S. thesis MIT. department of Electrical Engineering and Computer Science.

Fitz, Kelly and Bill Walker. "Lemur." Lemur Documentation 1-6.

Fitz, Kelly 1992. "Time and Frequency Scale Modification of Audio Signals Using an Extended Sinusoidal Model." M.S. thesis. University of Illinois. department Electrical Engineering.

Fitz, Kelly, William Walker, and Lippold Haken 1992. "Extending the McAulay-Quatieri Analysis for Synthesis with a Limited Number of Oscillators." *Proceedings of the 1992 International Computer Music Conference.* San Jose: Computer Music Association. 381-382

Freedman, M.D. 1967. "Analysis of Musical Instrument Tones." *Journal of the Acoustical Society of America.* 41: 793-806.

Freedman, M.D. 1968. "A Method for Analyzing Musical Tones." *Journal of Audio Engineering Society.* 16(4): 416-425.

Gordon, John W., John Strawn. 1985. " An Introduction to the Phase Vocoder." *Digital Audio Signal Processing: An Anthology.* John Strawn, ed. Los Altos, CA. William Kaufmann, Inc.

Grey, John M., John W. Gordon. 1978. "Perceptual Effects of spectral Modifications on Musical Timbres." *Journal Acoustical Society of America.* 63(5): 1493-1500.

Jaffe, David. 1987a. "Spectrum Analysis Tutorial, Part 1: The Discrete Fourier Transform." *Computer Music Journal.* 11(2): 9-24.

Jaffe, David. 1987b. "Spectrum Analysis Tutorial, Part 2: Properties and Applications of the Discrete Fourier Transform." *Computer Music Journal.* 11(3): 17-35.

Lansky, Paul. 1989. "Compositional Applications of Linear Predictive Coding." *Current Directions in Computer Music Research.* Max Mathews and John Pierce ed. Cambridge, Massachusetts: The MIT Press. 5-8.

Makhoul, John 1975. "Linear Prediction: A tutorial Review." *Proceedings of the IEEE* 63(4): 561-580.

McAulay, R.J., and T.F. Quatieri. 1986. "Speech Analysis/Synthesis based on a Sinusoidal Representation." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34(4): 774-754.

McAulay, R.J., and T.F. Quatieri. 1986b. "Speech Transformations Based on a Sinusoidal Representation." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34(6): 1449-1464.

McAulay, R.J., and T.F. Quatieri. 1991. "Peak-to-RMS Reduction of Speech Based on a Sinusoidal Model." *IEEE Transactions on Signal Processing* 39(2): 273-288.

Morrer, James A. 1978. "The Use of the Phase Vocoder in Computer Music Applications." *Journal of the Acoustical Society of America*. 26(3): 42-45.

Phillips, G. M. 1968. "Algorithms for Piece wise Straight Line Approximation." *Computer Music Journal.* 11:211-212.

Pierce, J.R. 1966. "Attaining Consonance in Arbitrary Scales," *The Journal of the Acoustical Society of America*: 40, 249.

Pierce, J.R. and M. V. Mathews, 1969. "Control of Consonance and Dissonance with Nonharmonic Overtones", *Music by Computers:* 129-132 and enclosed recording.

Quatieri, T.F., and R.G. Danisewicz. 1990. "An Approach to Co-Channel Talker Interference Suppression Using a Sinusoidal Model for Speech." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 38(1): 56-69.

Scaletti, Carla. 1989. "The Kyma/Platypus Computer Music Workstation." *Computer Music Journal*. 13(2): 23-38.

Serra, Xavier. 1989. "A system for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition. Ph.D. dissertation. Department of Music Stanford University. Stanford CA.

Serra, Xavier and Julius Smith III 1990 "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition." *Computer Music Journal* 14(4): 12-24.

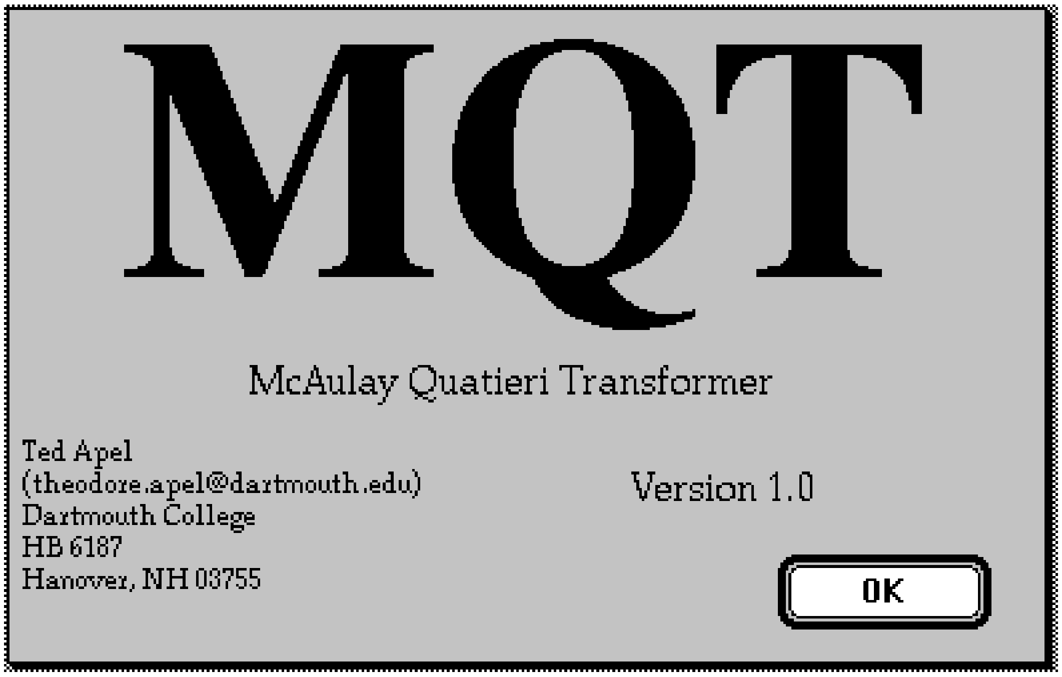
Slaymaker, Frank H. 1970. "Chords from Tones Having Stretched Partials." *The Journal of the Acoustical Society of America*: 1569-1571.

Smith, J.O., and X. Serra. 1987. "PARSHL: An Analysis/Synthesis Program for Non-harmonic Sounds based on a Sinusoidal Representation." *Proceedings of the 1987 International Computer Music Conference.* San Francisco: Computer Music Association. 290-297.

Strawn, John. 1980. "Approximation and Syntactic Analysis of Amplitude and Frequency Functions for Digital Sound Synthesis." *Computer Music Journal.* 4(3): 3-24.

Vercoe, Berry. 1990. "Csound: A manual for the Audio Processing System and Supporting Programs." MIT Media Lab.

Young, R. 1952. "Inharmonicity of Plain Piano Wires," *The Journal of the Acoustical Society of America*: 24, 267-273.


MQT

McAulay Quatieri Transformer

Ted Apel
(theodore.apel@dartmouth.edu)
Dartmouth College
HB 6187
Hanover, NH 03755

Version 1.0

User's Manual

Theodore Apel
June 1993
Bregman Electro-Acoustic Music Studio
Dartmouth College

MQT 1.0
© 1993 Ted Apel
Bregman Electro-Acoustic Music Studio
Dartmouth College
Hanover, NH 03755
theodore.apel@dartmouth.edu

Acknowledgments

Lemur was written by Bill Walker and Kelly Fitz of the CERL Sound Group at the University of Illinois.

Introduction

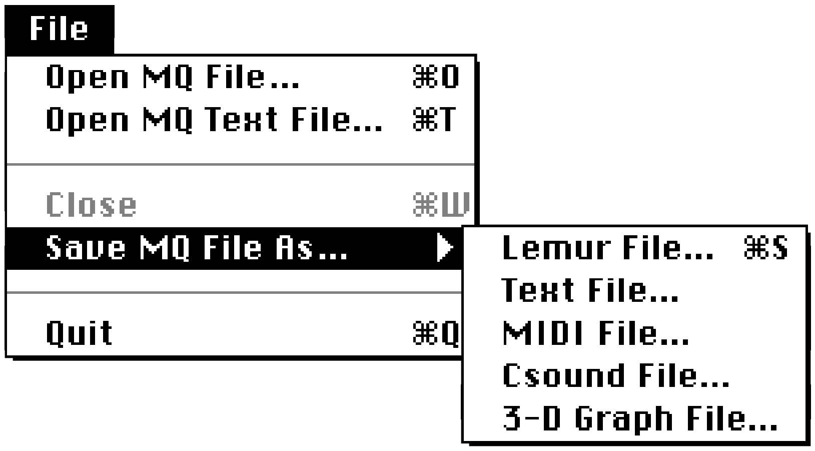
The purpose of MQT is to analyze and modify MQ files produced by the Lemur Analysis/Synthesis program. Analysis/synthesis programs are primarily used for two different types of transformations to a sampled sound. (i) Increasing or decreasing the length of a sound without altering the frequency, and (ii) increasing or decreasing the frequency of a sound without altering the length of the sound. These function are part of the Lemur analysis/synthesis program. There are many other types of transformations which are possible with analysis/synthesis programs, such as cross synthesizing, harmonizing, and MIDI conversion. The McAulay Quatieri analysis/synthesis method is well suited for such modifications because of its musically significant representation, inter frame peak tracking, and sturdy additive resynthesis methods.

McAulay Quatieri Analysis/Synthesis

The McAulay Quatieri analysis/synthesis method is based on a frequency domain representation of spectral peaks modeled on sinusoids. Unlike a simple Fast Fourier Transform (FFT), the MQ analysis/synthesis method saves only the amplitude, phase, and frequency of the modeled sine waves.

For a detailed discussion of the McAulay Quatieri analysis/synthesis method see the Lemur manual.

File Menu



Open MQ File...

Opens MQ files of type 'MQAN'. These files are produced with the Lemur program. MQT will not open sound files or other types of files. After an MQ file is opened the MQ file information is displayed. The file is then ready for analysis or transformation.

Open MQ Text File...

Opens text files of type 'TEXT'. These files are produced by the Save MQ Text Files As... command in the File menu or with another program which produces text files. The text file will be converted to a MQ file when opened, this allows analysis and transformations to be performed on the file. This command allows text files not created in Lemur or MQT to be converted to the Lemur MQ format.

Close...
Closes any open MQ file.

Save MQ file as Lemur file...
Saves the open MQ file in the Lemur format.

Save MQ file as Text file...
Saves the open MQ file as a text file. The text file may be viewed and modified in another program.

Save MQ file as MIDI file...
Saves the open MQ file as a standard MIDI file. The MIDI file may be opened and modified in any sequencer which opens standard MIDI files.

Save MQ file as Csound File...
Saves the open MQ file as a Csound score file. The score file may be opened and modified in Csound.

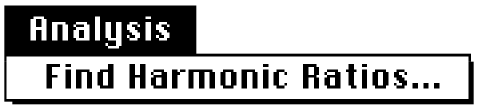
Save MQ file as 3D graph file...
Saves the open MQ file as a listing of Three parameters: frequency, amplitude, and time. This file may be opened in a 3-D graphing program for viewing.

Quit...
Quits the MQT program.

Edit Menu

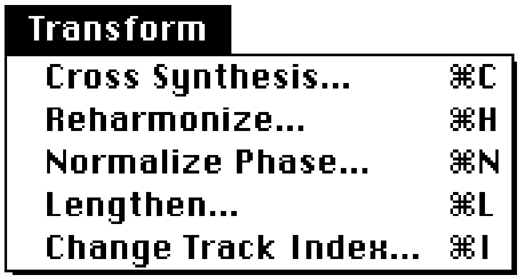
The edit functions are not used in MQT.

Analyze Menu

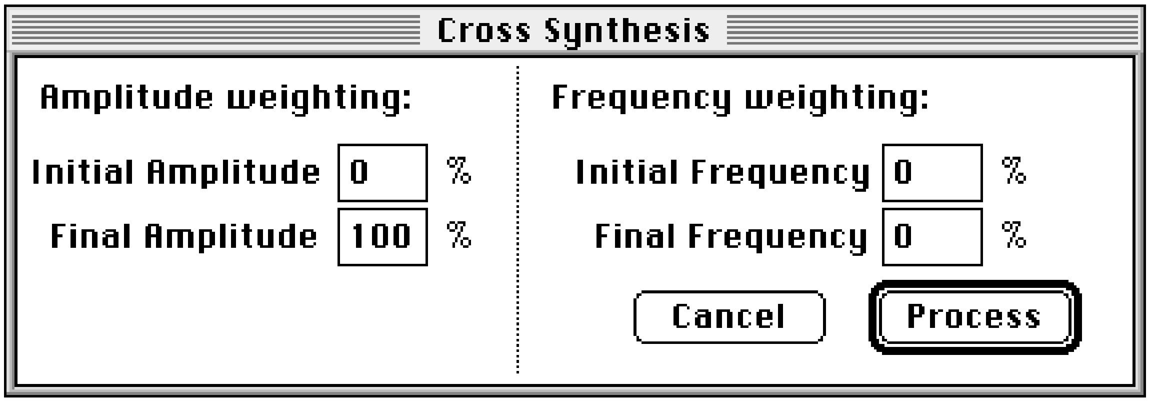


Find Harmonic Ratios...
The Find Harmonic Ratios command will analyze an MQ file for ratios between the fundamental and prominent harmonics in the file. These ratios are octave reduced and displayed as small integer ratios. These ratios may be used to build tuning systems from a sound.

Transform Menu



Cross Synthesis...


Cross Synthesis

Amplitude weighting:

Initial Amplitude 0 %

Final Amplitude 100 %

Frequency weighting:

Initial Frequency 0 %

Final Frequency 0 %

The Cross Synthesis command is used for combining two MQ files in various ways. The weighting of amplitude and frequency are selected in the Cross synthesis window. After **Cross Synthesis...** is selected, the weighting toward the open file is selected for both the amplitude and frequency at the beginning and end of the file. After the process button is pressed, the second MQ file is selected. After cross synthesis, the new file may be saved.

Reharmonize...
The reharmonize command is used to apply a pre defined spectra to a MQ file. All frequencies will be changed to match the frequencies of the Spectra file. The spectra file is a text file of up to 100 frequency values listed from low to high, each on a new line. When Reharmonize is selected, a spectra file is chosen to apply to the open MQ file.

Lengthen...
Lengthens the open MQ file by two times by reintroducing frames.

Track Index offset...
The Track Index command is used to change the connections between tracks. The user provides an integer number to increase or decrease the track indexes each frame. Tracks whose numbers increase beyond the number of tracks are reset.