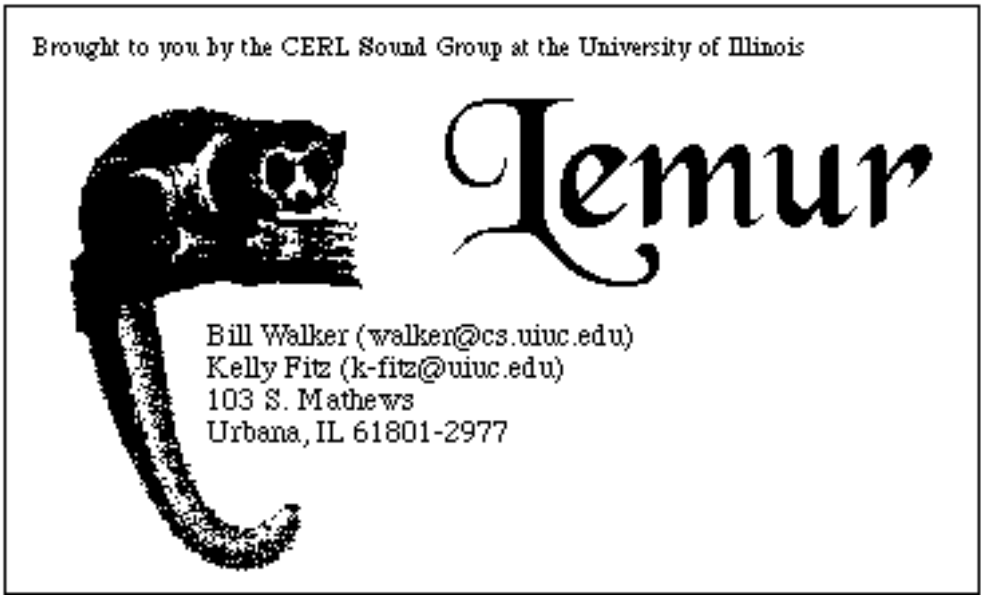

Acknowledgements

Lemur is based on `mqan`, a tool developed by Robert Maher and James Beauchamp at the University of Illinois.

Macintosh gurus—Kurt Hebel, John Brewer, Dan Walkowski

Sinusoidal modeling guru—Lippold Haken

Overview

Sinusoidal modeling

Lemur is based on the principal of sinusoidal modeling, especially the premise that any periodic signal can be reproduced by a summation of sine waves. During analysis, Lemur performs a series of Fast Fourier Transforms (FFT's) on a samples file to derive a set of time-varying sine waves. During synthesis, Lemur synthesizes these sine waves and adds them together to produce a new samples file.

McAulay-Quatieri technique

In a 1985 technical report from M. I. T. Lincoln Labs, McAulay and Quatieri proposed a sinusoidal analysis technique for speech processing. The basic premise of the MQ technique is that a sound can be represented by a collection of tracks, each of is a sine wave oscillator with time-varying amplitude and frequency. To construct these tracks, FFT's are performed on the signal being analyzed at regular intervals, called frames. Amplitude peaks in the resulting spectra are identified. These peaks are the most prominent frequencies in the sound at that instant. The peaks in adjacent frames are compared and matched. A continuous chain of these matches is a track. To ensure smooth tracks, the MQ analysis minimizes the difference between the frequencies of the peaks being matched. A peak that is not matched represents either the birth or death of a track.

Lemur extensions

Lemur provides some extensions to the basic McAulay-Quatieri technique.

Frequency Bins

The original MQ paper attempted to model psychoacoustic masking effects by suggesting that an amplitude threshold for peak detection should be based on the loudest peak in each frame. In other words, when the sound is loud, only loud peaks need to be represented, since quieter ones will be masked. When the sound is quiet, the quieter peaks are much more important. Unfortunately, this global threshold ignores the importance of frequency in masking effects. For example, a high frequency rarely masks a low one. Lemur provides a refinement of the original MQ amplitude threshold by breaking the frequency domain into logarithmically-sized bins. The loudest peak in each bin is determined, and an amplitude threshold for each bin is based on its loudest peak. This allows quiet peaks to be ignored in a bin containing loud peaks, while detecting quiet peaks in a bin without loud peaks.

Dormancy

In examining the results of an MQ analysis, one often observes a track which dies out and another track which is born a few frames later at roughly the same frequency. These are best understood as two portions of the same track. To facilitate this representation, Lemur allows tracks to lie dormant for a given number of frames before dying out. A dormant track has zero amplitude, but participates in peak matching. When a dormant track is connected to a live peak, Lemur interpolates to the new frequency and amplitude.

Analysis

During analysis, Lemur analyzes an AIFF samples file and creates an MQ file. While the analysis is running, the graph window displays the tracks as they are created. The smaller window gives you an idea of how long you will have to wait.

Spectrum

The Spectrum dialog box allows you to control the Fast Fourier Transformation (FFT).

FFT LENGTH • In order to take advantage of various symmetries, the FFT size is always a power of two. FFT length is always a tradeoff: A larger FFT gives more frequency accuracy, but averages over a longer period of time, which means poorer representation of transients.

WINDOW LENGTH • The input to each FFT is a windowed set of samples from the input file. The tradeoffs discussed above for FFT length also apply here. The window length must be smaller than the FFT length.

KAISER WINDOW PARAMETER • Lemur uses a Kaiser windowing function. Those of you who are familiar with the Kaiser window may adjust its parameter, which controls the shape of the window. Most users will never touch this option.

HOP SIZE • Between frames, Lemur "hops" over by a given number of samples. The hop size is generally smaller than the FFT length, so that each sample is involved in several FFT's. This is especially important when you intend to time stretch during synthesis.

INPUT SCALING • Lemur automatically scales the input file assuming that it has a full volume range. You can specify another input scaling if you wish.

Peak Selection

The Peak Selection dialog box controls various aspects of peak selection and track formation.

NUMBER OF FREQUENCY BINS • Lemur divides the frequency spectrum generated by the FFT into a number of frequency bins, and performs peak detection on each bin separately. The bins are logarithmically-sized, with the smallest bins containing the lowest frequency peaks. For an 1024 point FFT, the following bin arrangements are possible:

# of bins	bin sizes
1	1024
2	512 + 512
3	256 + 256 + 512
4	128 + 128 + 256 + 512
5	64 + 64 + 128 + 256 + 512
6	32 + 32 + 64 + 128 + 256 + 512
7	16 + 16 + 32 + 64 +128 + 256 + 512
8	8 + 8 + 16 + 32 + 64 + 128 + 256 + 512
9	4 + 4 + 8 + 16 + 32 + 64 + 128 + 256 + 512

Each bin must contain at least four points of the frequency spectrum. A larger FFT size means more frequency resolution and allows more frequency bins.

THRESHOLD • In addition to the time-varying threshold for peak detection which was discussed above, Lemur imposes an additional, constant threshold. This constant threshold is meant to represent the noise floor of the signal. The quietest allowable peak is always determined by the maximum of the two thresholds.

RANGE • Lemur locates the loudest peak in each frequency bin and subtracts the Range from its amplitude to determine the peak detection threshold. Reducing the range generally means an overall reduction in the number of peaks, but can have a detrimental effect on the quality of the synthesized sound.

DORMANCY • This setting controls the number of frames a track can lie dormant before actually dying out. With larger dormancy settings, the analysis will proceed more slowly, as Lemur attempts to match the dormant tracks at each frame. A higher dormancy will also delay the initial appearance of the tracks in the graph window.

CAPTURE RANGE • Lemur imposes a neighborhood for peak matching which varies with frequency. The default value seems to work fine.

Synthesis

During Synthesis, Lemur examines an MQ file and produces a new AIFF samples file. While the synthesis is running, Lemur displays the tracks as they are read from the MQ file.

Scaling and shifting

One of the most important features of sinusoidal techniques is the ability to perform various transformations which are difficult in the time domain. Lemur provides three transformations during synthesis.

TIME SCALING is performed by adjusting the temporal spacing between frames. Unlike playing your 33 RPM records at 45 RPM, Time Scaling changes the duration of your sound without changing its pitch. If lengthening your sound produces strange artifacts, try re-analyzing with a smaller hop size.

FREQUENCY SCALING multiplies every frequency in the MQ file by a given value before synthesizing. This preserves frequency ratios, but does not model the fixed formants of speech. As a result, frequency scaled speech will not sound like the original speaker talking at a higher pitch. Note that frequency scaling does not change the duration of your sound. If you scale up your frequencies too far, some of then will fall above the Nyquist rate and cause aliasing.

FREQUENCY SHIFTING adds a given value to every frequency in the MQ file before synthesizing. This does not preserve frequency ratios, but (for small shifting values) tends to preserve fixed format structures.

Control files

The three transformations discussed above can be controlled in two different ways—by specifying constant values or by specifying a control file. When a control file is specified, Lemur reads a new value from the control file for each frame. Note that the control file must have the same duration as the original sample file. Thus, the control file provides time-varying control over shifting and scaling. You can construct these control files (which are ordinary AIFF samples files) using whatever tools you have, or with Lemur's simple breakpoint editor, which can generate files with step functions or linear segments.

Control files are interpreted in several ways, depending on what they are used to control. When a control file is used for time scaling, the samples in the control file are normalized to values between +1 and -1. The time scale is ten (10) raised to this power. Thus, you can expand or contract by a factor of ten. A sample value of zero has no effect. The same algorithm is used for frequency scaling, so you can multiply or divide your frequencies by a factor of ten. Again, a sample value of zero has no effect. When a control file is used for frequency shifting, the samples in the control file are normalized to plus or minus half the sample rate (also known as the Nyquist rate). A sample value of zero has no effect.

The control files facility should prove useful to composers seeking novel effects. Time-varying time scaling is also useful for research in speech processing. Constant time scaling does not model what happens when people speak slowly, since the consonants are stretched along with the vowels. If you have an algorithm which examines the MQ file and decides which parts are vowels and which parts are consonants, you can automatically construct a control file which will stretch the vowels while leaving the consonants untouched. Similar ideas might be used to stretch the sustain of a violin tone while leaving the attack transient untouched.

MQ file format

The MQ file produced by a Lemur analysis begins with a header which contains the following information:

2 bytes	integer	versionNumber
2 bytes	integer	headerLength
12 bytes	floating point	inputScaleFactor
12 bytes	floating point	analysisThreshold
12 bytes	floating point	analysisRange
2 bytes	signed integer	numberOfFrequencyBins
2 bytes	signed integer	FFTlength
2 bytes	signed integer	windowLength
2 bytes	signed integer	analysisHopSize
4 bytes	unsigned integer	originalNumSamples
12 bytes	floating point	sampleRate
12 bytes	floating point	captureRange
2 bytes	signed integer	trackDormancyPeriod
12 bytes	floating point	timeScale
12 bytes	floating point	frequencyScale
12 bytes	floating point	frequencyShift
12 bytes	floating point	windowControl
4 bytes	<reserved>	timeScaleFile
4 bytes	<reserved>	freqScaleFile
4 bytes	<reserved>	freqShiftFile

Following this header, the file contains successive frames in chronological order. At the beginning of each frame is a two byte integer which specifies how many peaks are in that frame. Empty frames are allowed. The peaks are presented in order from lowest frequency to highest. Each peak has the following format:

12 bytes	floating point	magnitude

12 bytes	floating point	frequency
12 bytes	floating point	phase
2 bytes	integer	nextPeakIndex

The nextPeakIndex is used to connect a peak with the next peak in its track by giving the ordinality of a peak in the next frame. For example, if a given peak has the value 5 for its nextPeakIndex, then that peak should be connected with the fifth peak in the next frame.

Known Problems

Lemur may behave strangely if the program is run from a volume mounted with AppleShare or if you attempt to analyze a file on a remote volume.

Lemur is based on code which was developed in a UNIX environment. Hence, it does not conform to the Macintosh programming paradigm of an “event loop” based application. Those of you who are familiar with the details of the Macintosh interface will notice a few irregularities. For example, selecting desk accessories from the apple menu does not always work, nor does

the System 7 application menu. We hope you will bear with these problems which we may try to fix in a future release.

Bibliography

T. F. Quatieri and R. J. McAulay, *Speech Analysis/Synthesis Based on a Sinusoidal Representation.* Technical Report 693, Lincoln Laboratory, M. I. T.

Robert Crawford Maher, *An Approach for the separation of voices in composite musical signals.* Ph. D. dissertation, April 1989. University of Illinois at Urbana-Champaign.

Xavier Serra, *A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition.* Department of Music Report No. STAN-M-58, Center for Computer Music Research in Music and Acoustics, Stanford University. (Originally an October, 1989 Ph. D. dissertation).

John Sciarabba, *Psychoacoustics in sound synthesis.* M. S. Thesis, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign

(also available through the CERL Sound Group).