The Princeton Sound Kitchen is a repository of open source software for computer music written by graduate students and professors of the computer science and music departments at Princeton University. This studio report describes some recent open source software that has been added to this repository. These projects include computer music tools developed at the Music Department of Princeton as well as audio synthesis and recognition tools developed at the Computer Science Department. All the software is open source, uses standard cross-platform free software and has been tested in a variety of operating system and hardware configurations.

1 Introduction

The music department of Princeton University has a long tradition of computer music activity. A variety of computer music software tools have been developed by professors and graduate students as part of their compositional interests. In addition, a large number of software tools supporting research in audio synthesis and recognition have been developed at the Computer Science Department of Princeton University.

The developed software tools can be downloaded from The Princeton Sound Kitchen, a repository of open source computer music software developed at Princeton University. This report describes some of the recent additions to these repositories. All the software is open source, uses standard cross-platform software and has been tested in a variety of operating system and hardware configurations. The Princeton Sound Kitchen can be accessed from:

www.music.princeton.edu/winham/PSK

All the described projects are under active development and are used for musical composition, audio research and teaching at Princeton University. It is the belief of the authors that distributing the software as open source will increase its usefulness to the computer music community and the generated feedback will help make the software better.

2 Java Feature Extraction Application

Tae Hong Park, Graduate Student, Music

The Java Feature Extraction Program is software written in pure Java for the purpose of extracting musical audio signal features using signal processing techniques in the frequency and time domain. A stand-alone Java Application programming approach for its implementation has been chosen to make system independence one of the primary objectives. This project originated at Dartmouth but has moved to Princeton where it is being updated, refined and new features are being added. The features that are included in the current program are all "home-brewed" algorithms. They have all been implemented from scratch whenever possible for the purpose of learning details rather than implementing them from already existing papers, programs or code. However, they are hardly new nor ground breaking discoveries but rather serve as a basis for understanding feature spaces of musical audio signals. Figure 1 shows a screenshot of the system. More information about the software can be found at (Park 2000) which can be downloaded from: www.music.princeton.edu/~park/thesis/dartmouth/main.html
The algorithms and features in the program are divided into frequency and time domain analyses. The frequency domain features are based on the FFT and are analysis and display of a single FFT frame (Fast Fourier Transform), Peak Tracking, Spectral Centroid and Spectral Smoothness which use the Short-Time Fourier Transform. In the time domain current algorithms are Noise Content Analysis (LPC residual method), Pitch Tracking (using autocorrelation, interpolation and moving average filter of autocorrelation peaks), DC offset removal, Amplitude Envelope, Attack Time measurement, Amplitude Modulation Frequency LFO determination and Pitch modulation LFO determination. All analysis algorithms offer various parameters pertinent to each situation such as window type, percent overlap, resolution that aid in the acquisition of data according to one’s needs. Other basic functions include reading *.au, *.wav and *.aiff files, displaying extracted features with zoom in/out capabilities and standard play/record features of audio signals. The software as mentioned above has been implemented in pure Java and does not use any native code methods and can be run using Java 2 SDK v1.3 or higher and Java Runtime Environment Standard Edition v1.3 (JRE) or higher (http://java.sun.com).

3 MARSYAS

George Tzanetakis, Graduate Student, Computer Science

MARSYAS (MusicAl Research SYstem for Analysis and Synthesis) is a software framework, written in C++, for rapid prototyping of audio recognition. In addition, a graphical user interface for browsing and editing large collections of audio files, written in JAVA, is provided. The primary motivation behind MARSYAS has been research in content-based audio information extraction tools.

The core of MARSYAS is short time audio feature extraction. The available feature families are based on the following analysis techniques: Short Time Fourier Transform (STFT), Liner Prediction Coefficients (LPC), Mel Frequency Cepstral Coefficients (MFCC), the MPEG analysis filterbank and Cochlear filterbank models.

Multiple-feature automatic segmentation and classification is supported. Classification schemes that have been evaluated are Music/Speech, Male/Female/Sports, 7 Music Genres (Classical, Country, Disco, Fuzak, Hip Hop, Jazz, Rock), Instruments and Sound Effects. In addition it is easy to create other classification schemes from new audio data-sets. The currently supported classifiers are Gaussian Mixture Model (GMM), Gaussian, K Nearest Neighbor (KNN) and K-means clustering. Similarity retrieval and thumbnailing based on automatic classification and segmentation are also supported. MARSYAS has been designed to be flexible and extensible. New features, classifiers and analysis techniques can be added to the system with minimum effort.

Several utilities for automatic and user evaluation of the developed algorithms are included in the system. Using these tools a series of user experiments in segmentation, thumbnailing and beat perception have been conducted and their results have been used to evaluate and inform the design of new computer audition algorithms.

Several different browsing and visualization 2D and 3D displays are supported. All these interfaces are informed by the results of the feature based analysis. Some examples are:

- A waveform editor enhanced with automatic segmentation and classification. The editor can be used for “intelligent” browsing and annotation.
- Timbregram: a static visualization of an audio file that reveals timbral similarity and periodicity using color.
- TimbreSpace: a 3D browsing space for working with large audio collections based on Principal Components Analysis (PCA) of the feature space (Figure 2).
- GenreGram: a real time 3D graphics visualization display showing the current mixture of classification decisions as rotating cylinders for each classification category (Figure 2).

The software follows a client-server architecture. All the signal processing and statistical pattern recognition algorithms are performed using a server written in C++. The code is optimized resulting in real-time feature calculation, analysis and visualization display updates. The use of standard C++ and JAVA makes the code easily portable to different operating systems. MARSYAS has been tested in various Unix and Windows configurations. A general overview of the system can be found at (Tzanetakis and Cook 2000b) and the software can be downloaded from: www.cs.princeton.edu/~gtzan/marsyas.html.
4 STK

Perry Cook, Professor, Computer Science

The Synthesis ToolKit (STK) in C++ was designed to provide a portable, object-oriented, user-extensible environment for real-time audio and sound synthesis development. The desire was to establish a framework for rapid experimentation, such as implementing “intelligent player” objects, or use of custom controllers coupled to new synthesis algorithms. Also, for research, teaching, composition, and performance, there was a desire to create a set of examples of different synthesis techniques which wherever possible share a common interface, but allow the unique features of each particular synthesis algorithm to be exploited. By sharing a common interface, algorithms can be rapidly compared. In addition, synthesis can be accomplished in a scaleable fashion, by selecting the algorithm that accomplishes a desired task in the most efficient and/or expressive manner.

Nearly all of STK is written in generic C and C++ and can be compiled on any system with a C++ compiler. Cross-platform functionality is further aided by encapsulating operating system dependencies, such as real-time sound and MIDI input/output, within just a few classes. In keeping with cross-platform support and compatibility, simple Graphical User Interfaces for Synthesis ToolKit instruments have been implemented in Tcl/Tk. An example of one of these interfaces is shown in Figure 3. These allow simple real-time editing of the important control parameters of each synthesis algorithm using standard MIDI-type control messages (via SKINI), so MIDI controllers can be exchanged for GUI control enabling real-time expressive synthesis control.

All source code for the Synthesis ToolKit is made available freely for academic and research uses via various ftp servers, including Princeton Computer Science, the Princeton Sound Kitchen, and the Stanford Center for Computer Research in Music and Acoustics (CCRMA).

STK has also been enhanced, maintained, and supported by Gary Scavone of Stanford CCRMA. Numerous algorithms from STK have been ported to other sound synthesis systems such as the Vanilla Sound Server from NCSA, Csound, MSP, and SuperCollider. More information about STK can be found at (Cook and Scavone 1999) and at: www-ccrma.stanford.edu/software/stk

5 curvePainter

Colby Leider, Graduate Student, Music

“curvePainter” is graphical tool for composition with continuous control functions. Motivated by a desire to facilitate the composition of music for instruments and tape, both independently and collectively, curvePainter allows composers to “paint” musical parameters in the abstract and supply specific mappings to musical parameters in the target synthesis or notation environment at a later time.

The program generates text files of floating-point numbers for export to other software sound synthesis environments, such as STK (SKINI), SuperCollider, Max/MSP, RTcmix, OpenMusic, and Csound. The user can create, view, and edit up to twenty-four independent two-dimensional vectors of floating-point data ("curves") using a variety of generative techniques. Additionally, curves may be extracted from various characteristics of imported sound sets, such as their spectral centroid, amplitude envelope, and spectral peaks.

Many processing functions are supplied that allow the user to shape individual curves, including simple filters, a combander, and a phase shifter. Furthermore, curves may "morph" into each other to create new curves, or several curves may be averaged together.

Because the program makes no assumptions about the target environment, files generated with curvePainter can be used to control any parameter in any kind of synthesis model. Curves might be used to control the time-varying amplitudes of harmonics in an additive synthesis instrument; the density, duration, and frequency of grains in a granular synthesis instrument; the trajectories of point sources in a spatialization instrument; or the breath pressure, embouchure, pitch, and vibrato of a flute physical model. The curves may also of course be used to create deterministic or stochastic score files. curvePainter has been described in (Leider 2000; Leider and K.Burns 2000). Further information about the software can be downloaded from: music.princeton.edu/~colby
YAML

George Tzanetakis, Graduate student, Computer Science

YAML is an easily customizable and extensible environment for browsing, editing, generating and transforming high-level structured score files into lower level control score files. For now the input and output of YAML are SKINI files. SKINI files are an extended textual representation of MIDI messages (www.cs.princeton.edu/~prc).

The use of ML-Lex and ML-Yacc parsing tools facilitates the addition of parsers for other score files. Two interaction modes are supported. A graphical user interface based on a piano-roll representation is provided. The interface is written in JAVA and communicates with YAML via sockets. In addition, users can interact directly with the YAML environment using and extending the provided functions or adding their own. The use of higher-order functions and pattern-matching provided by SML makes various types of symbolic processing for score files easy to express. YAML has been developed with the help of Neophytos Michael who is a graduate student at the Computer Science Department at Princeton University.

PuCola is a 3D model of a soda can that is slid across various surface textures. The user controlled sliding speed and texture material result in the appropriate changes to the sound in real time. Similarly the Gear is a 3D model of a gear that is rotated using a slider. These programs are intended to be demonstration examples of how sound designers in virtual reality, augmented reality, games, and movie production can use parametric synthesis in creating believable real-time interactive sonic worlds. These 3D graphics interfaces were first described in (Tzanetakis and Cook 2000a).

References


7 Sounding Object Controllers

George Tzanetakis, Graduate student, Computer Science

Sound Object Controllers are a series of simple 3D graphics interfaces that have been written in Java3D with the common theme of providing a user-controlled animated object connected directly to the parameters of an audio synthesis algorithm. This work is part of the Physically Oriented Library of Interactive Sound Effects (PhOLISE) project (Cook 1999). This project uses physical and physically-motivated analysis and synthesis algorithms such as modal synthesis, banded waveguides, and stochastic particle models, to provide interactive parametric models of real-world sound effects. Figure 4 shows two of these sounding object controllers.

Figure 4: Gear and PuCola controllers.