MIR for audio signals using Marsyas-0.2

ISMIR 2006 Tutorial
Victoria, Canada
08.10.2006

George Tzanetakis (gtzan@cs.uvic.ca)
Luís Gustavo Martins (lmartins@inescporto.pt)
Summary

- Marsyas Overview
- Users & Applications
- Basic Mechanics
- Usage Scenarios

  · BREAK (15 min)

- Architecture
- Hands-on
- Interoperability
- Work in Progress
- Future Work
Marsyas Overview

- Software framework for audio analysis, synthesis and retrieval
  - Efficient and extensible framework design
    - specific emphasis on Music Information Retrieval (MIR)
    - C++, OOP
    - Multiplatform (Linux, MS Windows®, MacOSX®, ...)
  - Provides a variety of building blocks for performing common audio tasks:
    - soundfile IO, audio IO, signal processing and machine learning modules
    - blocks can be combined into data flow networks that can be modified and controlled dynamically while they process data in soft real-time.

Diagram:
- WAV source
- Hanning
- FFT
Marsyas Overview

- Marsyas Brief History
  - 1998 ~2000
    • Created by George Tzanetakis during his PhD activities at Princeton
  - 2000 ~2002
    • Marsyas 0.1
      - First stable revisions of the toolkit
      - Distributions hosted at SourceForge
      - Creation of a developer community
        » User and Developer Mailing lists
  - 2002 ~ …
    • Marsyas 0.2
      - Major framework revision
      - SourceForge SubVersion

http://www.cs.princeton.edu/~gtzan
http://sourceforge.net/projects/marsyas/
http://svn.sourceforge.net/viewvc/marsyas/?sortby=rev&sortdir=down
Users & Applications

- **Musicream** (Masataka Goto)
  - Music playback system with similarity capabilities
  - Uses Marsyas as its music similarity engine

http://staff.aist.go.jp/m.goto/Musicream/
• **SndPeak** (Princeton)

  - Uses Marsyas 0.1 for:
    - FFT magnitude spectrum
    - real-time spectral feature extraction
      - centroid
      - rms
      - flux
      - rolloff

http://www.cs.princeton.edu/sound/software/sndpeek/
Marsyas has been used at INESC Porto for the last 4 years...

**Audio Analysis Software prototypes:**
- Feature Extraction
- Audio segmentation/classification
- Audio fingerprinting
- Speaker Segmentation
- Music and Auditory Scene Analysis

http://www.inescporto.pt
http://www.inescporto.pt/~lmartins
Users and Applications

- Desert Island
  - Undergraduate at the Univ. Missouri
    - Kansas Jared Hoberock
    - Dan Kelly Ben Tietgen
Related Work Context

- **Open Source frameworks**
  - CLAM (http://clam.iua.upf.edu/)
  - STK (http://crrma.stanford.edu/software/stk/)
  - Chuck (http://chuck.cs.princeton.edu/)
  - PureData (Pd) (http://crca.ucsd.edu/~msp/software.html)
  - Open Sound Control (OSC) (http://cnmat.berkeley.edu/OpenSoundControl/)
  - FAUST (http://faudiostream.sourceforge.net/)

- **Commercial toolkits**
  - MAX/MSP® (http://www.cycling74.com/)
  - MATLAB® Simulink® (http://www.mathworks.com/products/simulink/)
  - LabView® (http://www.ni.com/labview/)
  - DirectShow® GraphEdit (http://windowssdk.msdn.microsoft.com/en-us/library/ms787460.aspx)
Audience Background?

- C++ ?
- MATLAB® ?
- Python™ ?
- WEKA ?
- Pd, MAX/MSP® ?

- DSP ?
- Symbolic Processing ?
- Machine Learning ?

- Linux ?
- MacOSX® ?
- Windows® ?

- Cooking 😊 ?
Basic Mechanics

• **Downloading Marsyas**
  - **SourceForge.net**
    • *Stable* releases
    • SubVersion (SVN) development code
• Structure of the Marsyas Distribution

  - Main files:

    • INSTALL, COPYING, THANKS, README, AUTHORS, TODO
      - Text files with some important info

    • ChangeLog
      - Provides usefull info about the evolution of the software

    • configure.in, Makefile.am
      - Main files edited by the user/programmer required for autotools. Only these files are needed to be changed when adding new subdirs or config options to the distribution
Basic Mechanics

- **Structure of the Marsyas Distribution**
  - **Main Subdirectories:**
    - `./config`
      - Configuration files used by autotools
    - `./distributed`
      - Marsyas (experimental) classes for distributed processing
    - `./doc`
      - Assorted documentation about the Marsyas framework and applications
    - `./marsyas`
      - Main dir where all the important Marsyas code is. The source files inhere are compiled into a static lib that other programs can use to access Marsyas functionalities
    - `./marsyasMATLAB`
      - User MATLAB scripts (mfiles)
    - `./marsyasVisualStudio2003`
      - Project and Solution files for Visual Studio 2003
    - `./marsyasVisualStudio2005`
      - Project and Solution files for Visual Studio 2005
    - `./qt4GUIs`
      - Qt4® GUI application projects (.pro) that use Marsyas as its audio engine. README files are supplied with instructions on how to compile and run each project
    - `./src`
      - Sample executables main code that do some interesting things using the Marsyas software framework. Some are in fact intended to be used as actual research tools
Basic Mechanics

• **Linux Installation**
  - **System requirements**
    - gcc compiler (included in Linux distributions)
  - **Building**
    - As a root user
      ```
      > ./configure
      > make
      > make install  (as root user)
      >
      ```
    - No root privileges
      ```
      > ./configure -prefix=/home/username
      > make
      > make install
      >
      ```
Basic Mechanics

• MacOSX® Installation
  – System requirements
    • gcc compiler
    • XCode®
  – Building

As a root user
> ./configure
> make
> make install (as root user)
>
No root privileges
> ./configure -prefix=/home/username
> make
> make install
> 

http://www.apple.com/macosx/features/xcode/
Basic Mechanics

- Windows® (Cygwin) Installation
  - System requirements
    - Cygwin
  - Building

As a root user

> ./configure
> make
> make install (as root user)

No root privileges

> ./configure -prefix=/home/username
> make
> make install

http://www.cygwin.com/
Basic Mechanics

• Windows® Visual Studio® Installation

  – System requirements
    • Visual Studio® 2003/2005/Express2005 (free version)
    • Microsoft DirectX SDK® (free)
    • Microsoft Platform SDK® (free)

  – Building
    • Open a .sln or .vcproj file on Visual Studio®
    • Go to the “Build” menu…
      – … and select “Build Solution” (or just press F7)

http://msdn.microsoft.com/vstudio/
http://msdn.microsoft.com/directx/sdk/
Basic Mechanics

• Configure Options
  – Autoconf systems (Linux, MacOSX®, Cygwin)

> ./configure --help (see list of options)

– Allows building Marsyas with:
  • assertions enabled (-enable-assert)
  • debug support (-enable-debug)
  • MP3 support (libmad) (-enable-mad)
  • distributed support (-enable-distributed)
  • MATLAB® support (-enable-matlab)
  • Qt® support (-enable-qt)

> ./configure --enable-assert --enable-mad
> make
> make install (as root user)
**Basic Mechanics**

- **Configure Options**
  - **Visual Studio® (Windows®)**
    - just use the defined Project Configurations
      - *e.g.* Debug, Release, Debug_MATLABEngine, …
Usage Scenarios

• Marsyas command line tools
  - Demonstrate key capabilities of the framework
    • Some are actually research tools
  - Efficient and can execute in real-time
  - ANSI C++ only core
    • several optional libraries

  - Tools and examples:
    • sfplay
    • bextract
    • phasevocoder
    • sfplugin
    • ...
Usage Scenarios

- **Playing audio files**

  - E.g.: `sfplay`  

  ```
  > sfplay foo.wav
  > sfplay -s 10.0 -l 3.2 -r 2.5 -g 0.5 foo1.wav foo2.au -f output.wav
  > sfplay -l 3.0 foo.wav
  > sfplay foo.wav -p playback.mpl
  >
  ```

  - `s`  
    → start time for playback  
  - `l`  
    → length of playback  
  - `r`  
    → repeat times  
  - `g`  
    → volume (gain) value  
  - `p` *playback.mpl*  
    → save playback network as a `.mpl` plugin file
Usage Scenarios

• Extracting features from audio signals and training classifiers
  
  - E.g.: bextract

  > bextract -e STFTMFCC music.mf speech.mf -p ms.mpl -w myweka.arff
  >

  - STFTMFCC
    • extracts spectral and MFCC features
  - music.mf, speech.mf
    • lists of sound files (collections)
  - ms.mpl
    • “trained” Marsyas plug-in for realtime music/speech classification
  - myWeka.arff
    • WEKA file with extracted features
Usage Scenarios

- Marsyas plugins (.mpl files)
  - Allow to dynamically recreate a processing network in runtime

- Audio playback

  > sfplugin -p playback.mpl foo.wav
  >

- Realtime audio classification

  > sfplugin -p ms.mpl unknownAudioSignal.wav
  >
Usage Scenarios

- Digital Signal Processing

-E.g.: phasevocoder

> phasevocoder -p 1.4 -s 100
BREAK TIME! 😊

15 Minutes Break…
- **Marsyas 0.2**
  - New **dataflow model** of audio computation

  - hierarchical messaging system used to control the dataflow network (inspired on *Open Sound Control (OSC)*)
  - **general matrices** instead of 1-D arrays as data

Marsyas::IMAGE PROCESSING?!?
Architecture

- **MarSystem Slices**
  - Separating things that happen at the same time from things that happen in different times

**Marsyas 0.1**

- 512 values
- FFT
- 512 values

**Marsyas 0.2**

- 22050 Hz
- 512 smpls * 1 obs
- FFT
- 22050 / 512 Hz
- 1 smpl * 512 obs

**Correct semantics for spectral processing**
Implicit Patching VS Explicit Patching

**EXPLICIT PATCHING**

\[
\text{Source} \rightarrow F1 \rightarrow F2 \rightarrow F3 \rightarrow \text{Destination}
\]

```c
// EXPLICIT PATCHING
source, F1, F2, F3, destination
# connect the appropriate in/out ports
connect(source, F1);
connect(source, F2);
connect(source, F3);
connect(F1, destination);
connect(F2, destination);
connect(F3, destination);
```

**IMPLICIT PATCHING**

\[
\text{Series} \rightarrow \text{Fanout} \rightarrow F1 \rightarrow F2 \rightarrow F3 \rightarrow \text{Destination}
\]

```c
// IMPLICIT PATCHING
source, F1, F2, F3, destination
Fanout(F1, F2, F3)
Series(source, Fanout, destination);
```
• **MarSystem Composites**
  - Series
  - Fanout
  - Fanin
  - Parallel
  - Accumulator
  - …

---

08.10.2006 Open Source Software Framework for Multimedia 30
Implicit VS Explicit Patching ➔ Neural Network

#IMPLICIT PATCHING
fanoutLayer1(ANN_Node11, ..., ANN_Node1N)
...
fanoutLayerM(ANN_NodeM1, ..., ANN_NodeMN)
ANN_Series(fanoutLayer1, ..., fanoutLayerM)
• **Typical feature extraction**

  - Time-frequency representation
  - Frequency summarization (MFCC)
  - Time summarization (Texture features)
Feature Extraction using Implicit Patching

- Series 1
  - Fanout 1
    - Zero Crossings
  - Series 2
    - Fanout 2
      - Spectrum
      - Centroid
      - Rolloff
      - Flux
  - Fanout 3
    - Texture Memory
      - Mean
      - Variance
  - Classifier

Source
Feature Extraction using Implicit Patching

```cpp
MarSystemManager mng;
MarSystem* Series1 = mng.create("Series", "Series1");
MarSystem* Fanout1 = mng.create("Fanout", "Fanout1");
MarSystem* Series2 = mng.create("Series", "Series2");
MarSystem* Fanout2 = mng.create("Fanout", "Fanout2");
MarSystem* Fanout3 = mng.create("Fanout", "Fanout3");
Fanout3->addMarSystem(mng.create("Mean", "Mean"));
Fanout3->addMarSystem(mng.create("Variance", "Variance"));
Fanout2->addMarSystem(mng.create("Centroid", "Centroid"));
Fanout2->addMarSystem(mng.create("RollOff", "RollOff"));
Fanout2->addMarSystem(mng.create("Flux", "Flux"));
Series2->addMarSystem(mng.create("Spectrum", "Spectrum"));
Series2->addMarSystem(Fanout2);
Fanout1->addMarSystem(mng.create("ZeroCrossings", "ZeroCrossings"));
Fanout1->addMarSystem(Series2);
Series1->addMarSystem(mng.create("SoundFileSource", "Source"));
Series1->addMarSystem(Fanout1);
Series1->addMarSystem(mng.create("Memory", "TextureMemory"));
Series1->addMarSystem(Fanout3);
Series1->addMarSystem(mng.create("classifier", "Classifier"));
```
• Implicit Patching and Filter Banks
  - Number of bands can be changed on-the-fly without any code recomplitation or user memory reallocation

Adapts automatically to any new number of filter bands
Hands-on

- Writing a new MarSystem
  - “Reverse” MarSystem

- Assembling a network
  - Sources & Sinks let data pass from input to output
  - Source → Gain → Reverse → SoundFileSink → AudioSink
Interoperability

• Marsyas Audio and MIDI I/O

  - RtAudio
    • Multiplatform C++ API for realtime audio input/output
      - Linux (native ALSA, JACK, and OSS)
      - MacOSX®
      - Windows® (DirectSound® and ASIO®)
      - SGI®

  - RtMIDI
    • Multiplatform C++ API for realtime MIDI input/output
      - Linux (ALSA)
      - MacOSX®
      - Windows® (Multimedia Library)
      - SGI®

Interoperability

- **Marsyas (↔) WEKA** (Data Mining Software in Java)
  - Marsyas already includes some machine learning blocks
  - Marsyas outputs extracted features as .arff files (WEKA)
    - features can be opened in WEKA for further evaluation and data modeling
Interoperability

- **Calling MATLAB® from C++ Marsyas code:**
  - MATLAB® engine API
    - exchange data (i.e. matrices) in run-time between C++ and MATLAB®
    - remotely execute commands in MATLAB® from a C++ routine
      - Access to all MATLAB® toolboxes, algorithms and available routines
      - Algorithmic validation of C++ routines
      - Quick and easy evaluation of proof of concepts
      - May not allow real-time operation…
        » Not such a big problem when evaluating or developing algorithms
  - Marsyas::MATLABEngine class
    - Utility class
      - Wraps MATLAB® engine calls for most POD types and Marsyas data types
      - Easy to send/receive data to/from MATLAB® from anywhere in the code

Interoperability

- **Python™ Bindings**
  - easily create scripts for rapid testing and prototyping of data flow networks
    - would require much more development effort in C++
    - bonus: no compiling overheads
  - can also be embedded in C++ code, similarly to MATLAB® (TBD)
    - less tools for signal processing in general, but can be used for many other purposes (“batteries included”)
    - less licensing headaches

http://www.python.org
Interoperability

- Marsyas and Trolltech Qt4®
  - Qt® Core features *optionally* used by Marsyas
    - Multi-platform *signal/slot architecture*
    - Multi-platform *threads → multithreaded processing*
    - Multi-platform *database access*
    - Multi-platform *XML I/O*
  - Qt® GUI Features optionally used by Marsyas
    - Multi-platform *Widgets*
    - Multi-platform *OpenGL*

Qt4® is available as GPL open source code for all platforms

http://www.trolltech.com
• Marsyas GUI
  - Allows rapid algorithm development and debugging
  - GUIs for realtime interaction with processing blocks
  - Visualization tools
    - Data plots, controls, ...
  - MATLAB® scripts
    - create, load, save mfiles
    - run scripts
  - Deployment
    - Developed algorithms can be saved as a .mpl file and run as a console app
Work in Progress

• **Data Plotting GUIs**
  - For efficient and easy data inspection and manipulation

• **Machine Learning algorithms**
  - Marsyas 0.1 → Marsyas 0.2
  - Implementation/integration of new C++ algorithms

• **Auditory Scene Analysis**
  - New analysis front ends
    - Sinusoidal modeling
    - Auditory models
Work in Progress

- Marsyas:
  - Software framework for audio analysis, synthesis and retrieval
• Image, Video and Multimodal Analysis

  - Core modifications are being made to allow seamless processing of both visual and audio signals in a same data flow network
    - Marsyas general matrices are used to store and deliver image data
  
  - No modules are available so a lot of effort needs to be invested in order to create or port visual processing algorithms
    - Implementation will be made in a “as-needed” basis
  
  - Combined analysis and processing of audio and video streams
Work in Progress

- **Distributed Processing**
  - Marsyas 0.2 already includes some (experimental) routines for distributing processing blocks over remote machines.

Distributed audio feature extraction

Realtime distributed harmonizer
Flexible scheduling for dataflow audio processing

- Multiple timers
  - realtime, beat-time, virtual-time

- Multiple events
  - updateControl, patch(control1, control2,…), expression
Future Work

• Distributed Processing
  – Current implementation
    • Only tested on Linux
      → Port and test on MS Windows®/MacOSX® platforms
    • Uses static machine addressing
      – i.e. must define IP address of remote machine
        » Not very flexible
        » ...
  • Dynamic Ad-Hoc approach?
    – Marsyas would be able to automatically (re-)distribute processing blocks over remote machines as they appear/disappear from the network...
Future Work *(we need your Help!)*

- SDIF output I/O
- XML Marsyas schemas
- Visual Network Builder
- Tool specific GUIs
- Marsyas $\leftrightarrow$ VST® plugin
- Marsyas $\leftrightarrow$ Pd and MAX/MSP® external
- Control performance capturing and playback
- More sound synthesis blocks (port from STK)

- Implement every algorithm for Audio MIR published in ISMIR! 😊
Thank you!

(Lifetime) Future work...

gtzan@cs.uvic.ca
lmartins@inescporto.pt

With the kind support of:

08.10.2006