Signal Analysis and Feature Extraction for MIR Applications

- What do we want to do?
  - Match, search, index, transcribe, src-sep, ...
- What do we need to know to do it?
  - Basic feature set
  - Higher-level features
  - Feature data post-processing
  - Application integration
- MIR application design
  - Many are not “IR” at all
  - How does the metadata fit in?
- Feature vector design for applications

Problem Statement: Applications

- Examples
  - Automatic playlist generation
  - Audio transcription
Music Metadata

• Introduction
  – Kinds of Audio Data and Metadata
  – Dimensions of Music Info. Retrieval
  – APIs for MIR Tools
• Multimedia Databases
  – Feature Vectors and Indexing
  – Feature Extraction and Signal Analysis
  – Numerical Processing: Clustering, Classification
• Audio Signal Processing for Feature Extraction
  – Time Sequences, Windowing
  – Analysis Domains, Transformations
  – Multi-level, High-level
• Data Smoothing and Reduction

Signal Analysis

• Time–domain Audio Analysis
  – Windowed RMS Envelope Extraction
  – Beat Detection and Tempo Analysis
  – Time–based signal segmentation
• Frequency–domain Analysis
  – Pitch Detection Techniques
  – Spectral Analysis and Interpretation
  – Spectral Peaks and Tracking
  – Other Spectral Measures
• Other Kinds of Analysis: Wavelets
• Cross–domain analysis

Numerical Processing

• Data Reduction, Smoothing
• Correlation, Grouping
• Princ./Indep. Component Analysis
• Audio Segmentation and Musical Form
• Clustering and Classification

Databases & Applications

• Database Issues
• Handling of Large or Dynamic Feature Vectors
• Application Requirements and Design
• Searching, Indexing, and Players
• Audio Summarization and Thumb–nailing
• Content Matching and Finger–printing
• Data Clustering and Genre Classification
• Other Applications: Mapping Systems

Typical Processing Stages

• Input processing
  – Streaming, decompression, reformatting
• Signal segmentation, windowing
  – window size, share, overlap
• 1st–pass windowed feature extraction
  – Basic time–, freq–domain features
• 2nd–pass feature processing
  – Feature massaging, smoothing, pruning
  – 2nd–pass features (tempo, segmentation)
• Post–processing, data output
  – Many options

MIR Application Design

• Dimensions
  – Content format
  – Low–level analysis procedures
  – High–level derived features
  – DB design
  – Application flow and integration
• Design Issues
  – System architecture and design impacted by each of the MDB dimensions
### Content Format
- Impacts all levels of system
  - Data volume, storage options, analysis DSP, DB design, etc.
- Systems may or may not maintain original source content (vs. metadata)
- Systems may preserve several formats of source and metadata (n-tier)
- This is typically a given rather than a design option

### Content Formats
- Audio-based
  - Properties/volume of source recordings
  - MP3/AAC/WMA decoders
- MIDI-based
  - Problems with MIDI, assumptions to make
  - Human-performed vs “dead pan” MIDI
- Score image based
  - Useful, but not treated here
- Formal language-based
  - SCORE, SMDL, Smoke, etc.
  - MusicXML

### Real Applications
- DBMS issues
- Query systems, browsers, and MIR frameworks
- Informed tools
- Stand-alone delivery applications

### Applications
- One-step Tools
  - Tracker, segmenter, single feature extraction
  - Interactive programs
- Multi-feature tools
  - Finger-print, thumb-nail, etc.
- Heuristic techniques
  - Blackboard, neural nets, SOMaps
- Real-world MIR applications

### Typke's Appl. Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Audio</th>
<th>Video</th>
<th>Textual</th>
<th>Score</th>
<th>MIDI</th>
<th>Image</th>
<th>Formal Lang.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Blips</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chords</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outlines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUI MI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIDI Generator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score images</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score Objects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio Blips</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chords</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outlines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUI MI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIDI Generator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score images</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score Objects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### MIR Systems & Retrieval Tasks
- Copyright and royalties
- Plagiarism
- Recommendation
- Sounds as
- Genre
- Mood
- Style
- Emotion
- Performance
- Feature
- Intertextuality
- Composer
MIR/MDB Applications

GT's “MIR Pipeline”

Dimensions of Music Information Retrieval Applications
• Indexing, query, access
  - Use content or metadata for query
• Understanding, transcription
  - Derive (music/speech) model
• Clustering, classification
  - Feature vector for discrimination
• Content identification, finger-printing
• Preference-matching, recommendation

Time Sequences, Windowing
• Read audio input
• Vector multiply by window function
• Perform analysis
• Step to next window
• Hop size not normally = window size
• Window features
  - Main lobe width, side lobe level, side lobe slope

Windows and their Spectra (see MAT 240B)
• Trade-offs between window characteristics
• Different windows for different analysis domains

Advanced Windows for Spectral Analysis
Windowing and the STFT

Multi-window Multi-rate Analysis
• Example: FMAK3 analysis driver
  • `-r rmsWindow_size rmsHop_size`
    - window size and hop size for the RMS time-domain analysis
  • `-f fftWindow_size fftLen fftHop_size`
    - for the FFT spectral analysis
  • `-l lpcWindow_size lpcOrder lpcHop_size`
    - for the LPC analysis
  • `-w fwtWindow_size fwtLen fwtHop_size`
    - for the wavelet analysis

Time-domain Audio Analysis and Applications
• Use rectangular window if no overlap or triangular window if overlapping
• Medium-sized window (10 Hz or better resolution desired)
• Derived windowed RMS value
• Count zero crossings

Windowed RMS Envelope Extraction
• C code for envelope extraction
  • Outer loop for windows
  • Inner loop to run window and compute RMS value
  • Silence threshold (noise gate)
  • Note-on trigger (peak detector)
  • Example sound: piano sample, drum loop

Optional Time-domain Steps
• Pre-filter to get low-freq and high-freq RMS values
• Process stereo channels to get M/S (sum/difference) signals
• Noise detection
• Silence detection
• Loop code examples and main()s

Feature-vector Design
• http://www.create.ucsb.edu/~stp/PostScript/PopeHolmKouznetsov_icmc2.pdf
• Application Requirements
  • Labeling, segmentation, etc.
  • Derive feature vector from the app requirements
• Kinds/Domains of Features
  • Time-domain
    • Simple features, onset detection
  • Frequency-domain
    • Spectrum, spectral statistics
    • Pitch, chroma, key
Feature Vectors and Indexing

- Feature = derived (numerical) parameter
- Feature vector = list of features for a single point/window in time, or average for an entire selection
- Feature table = list of feature vectors for several time slices (not always used)

Example Features

- Features:
  - Time-domain, low-level
    - Windowed RMS amplitude
  - Time-domain, high-level
    - Tempo, beat structure, segmentation
  - Frequency-domain, low-level
    - Pitch, spectrum, spectral peaks
  - Frequency-domain, high-level
    - Peak track birth/death statistics, instrument ID
  - Many other possibilities (see below)

Feature Vector Examples

Example: FMAK3 Feature Table

```cpp
class FeatureTable {
public:
  // Data members (instance variables)
  float mTimestamp;  // When do I start?
  float mTimeDur;   // How long a time-span do I represent?

  // Time-domain features
  unsigned int mRMSWindowSize; // Size of RMS window
  FeatureDatum mRMS;  // Rectangular-windowed RMS amplitude
  FeatureDatum mPeak;  // Max sample amplitude
  FeatureDatum mLPRMS; // RMS amplitude of LP-filtered signal
  FeatureDatum mHPRMS; // RMS amplitude of HP-filtered signal
  size_t mZeroCrossings; // Count of zero crossings
  FeatureDatum mDynamicRange;  // RMS dynamic range of sub-windows
  FeatureDatum mPeakIndex; // RMS peak sub-window index
  FeatureDatum mTempo; // RMS/FWT instantaneous tempo estimate
  FeatureDatum mTimeSignature; // Time signature guess
  unsigned int mBassNote; // Bass note (MIDI key number) guess
  FeatureDatum mBassDynamicity; // Bass note dynamicity (size of histogram)

  // Spatial features
  FeatureDatum mStereoWidth; // L/R difference
  FeatureDatum mSurroundDepth; // Front/Surround difference
  FeatureDatum mCenterDistinction; // Center vs. L/R sum difference

  // Frequency-domain features
  unsigned int mFFTWindowSize; // Size of FFT window
  FtVector mSpectrum;  // Hanning windowed FFT data (1024 points, or NULL)
  FtVector mReducedSpectrum; // 1-octave FFT data (10-12 points)
  FtVector mBandSpectrum; // 2.5-octave FFT data (4 points -- spectral bands)
  FPartialVector mSpectralPeaks;// List of major spectral peak indeces
  FPartialVector mSpectralTracks; // List of tracked peak frequencies
  FeatureDatum mSpectralCentroid; // Spectral centroid measure
  FeatureDatum mSpectralSlope; // Spectral slope measure
  FeatureDatum mSpectralVariety;// Inter-frame spectral variety measure

  // Hi-frequency properties
  FeatureDatum HiFreqBalance; // Relative HF level
  FeatureDatum HiFreqVariety; // HF inter-frame spectral variety
  FeatureDatum HiFreqCorrelation;// Correlation between HF and audio-band tracks

  // LPC features
  unsigned int mLPCWindowSize; // Size of LPC window
  FPartialVector mLPCFormants; // List of LPC formant peaks
  FPartialVector mLPCTracks; // List of tracked LPC formants
  FeatureDatum mLPCResidual; // LPC residual level (noiselessness)
  FeatureDatum mLTrackBirths;  // LPC formant peak track births, deaths

  // Wavelet-domain (FWT) features
  FtVector mWaveletCoeff; // Wavelet-projected coefficients
  FtVector mWTNSpectrum; // Reduced FWT HiFreq noise spectrum
  FtVector mWTTracks;  // List of tracked FWT peaks
  FeatureDatum mWTNoise; // FWT noise estimate
};
```

Example: FMAK3 Feature Table, cont’d

```cpp
// Frequency-domain features
unsigned int mFFTWindowSize; // Size of FFT window
FVector mSpectrum;  // Hanning windowed FFT data (1024 points, or NULL)
FVector mReducedSpectrum; // 1-octave FFT data (10-12 points)
FVector mBandSpectrum; // 2.5-octave FFT data (4 points -- spectral bands)
FPartialVector mSpectralPeaks;// List of major spectral peak indeces
FPartialVector mSpectralTracks; // List of tracked peak frequencies
FeatureDatum mSpectralCentroid; // Spectral centroid measure
FeatureDatum mSpectralSlope; // Spectral slope measure
FeatureDatum mSpectralVariety;// Inter-frame spectral variety measure

// Hi-frequency properties
FeatureDatum HiFreqBalance; // Relative HF level
FeatureDatum HiFreqVariety; // HF inter-frame spectral variety
FeatureDatum HiFreqCorrelation;// Correlation between HF and audio-band tracks

// LPC features
unsigned int mLPCWindowSize; // Size of LPC window
FPartialVector mLPCFormants; // List of LPC formant peaks
FPartialVector mLPCTracks; // List of tracked LPC formants
FeatureDatum mLPCResidual; // LPC residual level (noiselessness)
FeatureDatum mLTrackBirths;  // LPC formant peak track births, deaths

// Wavelet-domain (FWT) features
FVector mWaveletCoeff; // Wavelet-projected coefficients
FVector mWTNSpectrum; // Reduced FWT HiFreq noise spectrum
FVector mWTTracks;  // List of tracked FWT peaks
FeatureDatum mWTNoise; // FWT noise estimate
```

Windowed Feature Comparison

```cpp
class FeatureTable {
public:
  // Data members (instance variables)
  float mTimestamp;  // When do I start?
  float mTimeDur;   // How long a time-span do I represent?

  // Time-domain features
  unsigned int mRMSWindowSize; // Size of RMS window
  FeatureDatum mRMS;  // Rectangular-windowed RMS amplitude
  FeatureDatum mPeak;  // Max sample amplitude
  FeatureDatum mLPRMS; // RMS amplitude of LP-filtered signal
  FeatureDatum mHPRMS; // RMS amplitude of HP-filtered signal
  size_t mZeroCrossings; // Count of zero crossings
  FeatureDatum mDynamicRange;  // RMS dynamic range of sub-windows
  FeatureDatum mPeakIndex; // RMS peak sub-window index
  FeatureDatum mTempo; // RMS/FWT instantaneous tempo estimate
  FeatureDatum mTimeSignature; // Time signature guess
  unsigned int mBassNote; // Bass note (MIDI key number) guess
  FeatureDatum mBassDynamicity; // Bass note dynamicity (size of histogram)

  // Spatial features
  FeatureDatum mStereoWidth; // L/R difference
  FeatureDatum mSurroundDepth; // Front/Surround difference
  FeatureDatum mCenterDistinction; // Center vs. L/R sum difference

  // Frequency-domain features
  unsigned int mFFTWindowSize; // Size of FFT window
  FtVector mSpectrum;  // Hanning windowed FFT data (1024 points, or NULL)
  FtVector mReducedSpectrum; // 1-octave FFT data (10-12 points)
  FtVector mBandSpectrum; // 2.5-octave FFT data (4 points -- spectral bands)
  FPartialVector mSpectralPeaks;// List of major spectral peak indeces
  FPartialVector mSpectralTracks; // List of tracked peak frequencies
  FeatureDatum mSpectralCentroid; // Spectral centroid measure
  FeatureDatum mSpectralSlope; // Spectral slope measure
  FeatureDatum mSpectralVariety;// Inter-frame spectral variety measure

  // Hi-frequency properties
  FeatureDatum HiFreqBalance; // Relative HF level
  FeatureDatum HiFreqVariety; // HF inter-frame spectral variety
  FeatureDatum HiFreqCorrelation;// Correlation between HF and audio-band tracks

  // LPC features
  unsigned int mLPCWindowSize; // Size of LPC window
  FPartialVector mLPCFormants; // List of LPC formant peaks
  FPartialVector mLPCTracks; // List of tracked LPC formants
  FeatureDatum mLPCResidual; // LPC residual level (noiselessness)
  FeatureDatum mLTrackBirths;  // LPC formant peak track births, deaths

  // Wavelet-domain (FWT) features
  FVector mWaveletCoeff; // Wavelet-projected coefficients
  FVector mWTNSpectrum; // Reduced FWT HiFreq noise spectrum
  FVector mWTTracks;  // List of tracked FWT peaks
  FeatureDatum mWTNoise; // FWT noise estimate
};
```
Analysis Domains and Transformations

- Time-domain Audio Analysis and Applications
- Windowed RMS Envelope Extraction
- Beat Detection and Tempo Analysis
- Time-based signal segmentation
- Frequency-domain Analysis
- Pitch Detection Techniques
- Spectral Analysis and Interpretation
- Spectral Peaks and Tracking
- Other Spectral Measures
- Other Kinds of Analysis: Wavelets
- Cross-domain analysis

Time-domain Features

- RMS, Peak
- LPF/HPF RMS
  - e.g., F < 200 Hz, F > 2000 Hz
- Dynamic range
  - What window for calc?
- Zero-crossing rate (time- or freq-domain?)
- Higher-level statistics
  - Mean/variance
  - Variance of sliding windows
  - Spacing of peaks/troughs
  - Many other options
- Time-domain onset-detection & beats

Windowed Amplitude Envelopes

- Choice of window size, hop size, window function shape
- May use several frequency bands (kick drum vs. hi-hat)
- Useful for silence detection, beat tracking, simple segmentation, summarization, etc.
- Simple, effective, well-understood techniques, many options

Frequency-domain Features

- Spectrum, Spectral bins
  - Window/hop sizes
  - Improving spectral data: phase unwrapping, time realignment
- Spectral measures (statistical moments)
- MFCCs
- Peak-picking and peak-tracking
- Pitch-estimation and pitch-tracking

Frequency-domain Analysis

- Short-time Fourier transform
  - Configuration options and trade-offs
  - Interpretation/weighting of spectral bins (perceptual scales)
- Other frequency-domain techniques
  - Filter banks
  - Linear prediction
  - Filter matching
- Loads of options

Speech Spectrogram

- Kinds of spectral plots
- Features
The Pitch/Time Trade-off

Harmonics and Formants
- Source/Filter – instr resonances

Using FFT APIs
- Simple FFT
  - See MAT240B
  - See F. R. Moore’s Elements of Computer Music
- FFTW
  - FFTW data types
  - FFTW plans
  - See CSL Spectral class

Composite Spectra
- How to disambiguate?
- Track birth/death statistics
- Vibrato (see figure)
- Statistical techniques

Spectral Analysis and Interpretation
- Spectral data extraction
  - Base frequency
  - Overtone spectrum
  - Formants, resonances, regions
  - Instrument signatures
- Spectral statistics
  - Peak, mean, average, centroid, slope, etc.
  - Spectral variety, etc.

Spectra as Time-varying
- Track peaks/regions between frames (requires thresholds of change)
- Model the dynamicity (e.g., formant trajectory, vibrato extraction)
Spectral Peaks and Tracking
- Peak finding (remember autocorrelation?)
- Peak discrimination
- Peak continuation: tracks and guides
- Derived statistics
- Problem cases

Peaks and Tracks
- Peak-finding
  - Thresholds, distances, heuristics
- Peak-continuation
  - Inter-frame distances and guides
  - Dropped frames and stretching
  - Track birth/death criteria

Spectral Peak-Tracking Example

Spectral Peak-Detection Algorithm
- From Blum et al. patent # 5,918,223

Spectral Smoothness Measure

Smoothed Spectrum Types
Deriving Spectral Bands

- Octave-band loops
  - outer loop - step size doubles every octave
  - inner loop - sums bins in range
- Weighted, non-linear bands

Filter-based Pitch Detection

- Simple adaptive process for single-frequency source with strong fundamental (i.e., many, but not all, instruments and voices)
- Easily implemented in analog circuitry
- Many variations

Auto-Correlation

- Slide a signal across itself, taking the vector product at each step
- This AC array has a peak at 0, and the period of the signal
- No peaks for noise

Harmonic Product Spectra

- Decimation of FFT spectra, summation, and spectral peak location
- Assumes overtones are significant, not that fundamental is

Pitch Detection Techniques

- Find the period of a “periodic” signal
  - First guess whether or not it’s periodic
- Simple techniques work for many signals
  - Zero-crossings (with direction, slope)
  - Autocorrelation (with range limitation)
- It’s hard to tell when they fail
  - Random data, silence
  - Octave over/under-tone errors

Harmonic Product Spectra

- Implementation
- Outer loop (octaves)
  - Scales copy of spectrum into buffer
- Inner loop
  - Take max or avg of sub-window?
- Post
  - first max > min_val
Mel–Freq Cepstral Coefficients

- **Steps:**
  - Signal
  - FT
  - Log magnitude
  - Phase unwrapping
  - FT (or DCT)

- **Name reversal**

- **Interpretations**
  - Quefrency
  - Mel–scale
  - Mel–scale filters

- Instead of AC, use FFT or DCT of PDS
- Leads to interesting statistics of higher–level spectral properties, see next section

MFCC Analysis

- **Analogy**
  - Start with log spectrum of mixed complex tones: several sets of related partial peaks
  - Take, e.g., the autocorr. of the FFT PDS
  - Warped frequencies of peaks correspond to fundamental frequencies of overtone series

Comparison With LPC (by Andrianakis & White)

<table>
<thead>
<tr>
<th>FFT PSD</th>
<th>13-pole LPC Spectrum</th>
<th>20 Mel Spectrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>Frequency [Hz]</td>
<td></td>
</tr>
<tr>
<td>Car</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Spatial–domain Features

- M/S Encoding (stereo sum & difference)
- Surround–sound processing
  - L/R vs C
  - L/R vs Ls/Rs
- Frequency–dependent spatial separation
- Higher–dimensional sources
- Stem tracks

Other Feature Domains

- Other time–domain features
  - Beats, beat histograms
- Other frequency–domain features
  - Fluctuation patterns
- Other time–frequency transforms
  - Filter banks
- Wavelets
- Linear Predictive Coding

Review

- MIR Apps
- Signal analysis processing chains
- Feature vector design from app requirements
- Kinds of audio features
- Basic feature statistics
Beat-finding and Tempo Derivation

- Why?
  - Tempo and Beat are strong discriminators in judgements of music similarity, and even genre (Tzanetakis & Cooke 2002, Dixon et al. 2004).
  - Understanding the beat facilitates understanding the importance of other musical elements:
    - Relative importance of tonal features.
    - Diatonic or chromatic character of a piece.
    - Complexity of a piece.
  - Applications: musicology & ethnomusicology, automatic DJing, query by example, composition tools.

Modelling Rhythm

- "...the systematic patterning of sound in terms of timing, accent, and grouping." (Patel 2008 p.96)
  - (Not always periodic patterns)
  - Accent sources include: dynamics, melody, harmony, articulation, timbre, onset asynchrony etc.
  - Consists of hierarchical and figural (proximal) temporal structures.

Measuring Beat

- Inter-Onset Intervals (IOI)
- Inter-Beat Interval (IBI)
- Tempo: frequency of the beat (BPM)

Musical Time

- Multiple simultaneous levels of musical time
  - Tactus: the foot-tapping rate.
  - Tempo: estimated from tactus, typically median IBI.
  - Meter: Periodic perceived accentuation of beats.
  - Tatum: Shortest interval between events.
- Rubato – change in tempo during performance to emphasise structure.

Meter

- Meter is expressed in Western music as time-signatures (4/4, 3/4 etc).

(Courtesy Olivia Ladinig)
Rhythmic Strata
- Musical rhythm can be considered as composed of a hierarchy of temporal levels or strata (Yeston 1976, Lerdahl & Jackendoff 1983, Clarke 1987, Jones & Boltz 1989).

Hierarchical Grouping: Meter
- Meters are argued to arise from the interaction between temporal levels (Yeston 1976).
  - Therefore a meter implies two frequencies: the pulse rate and the measure (“bar”) rate.
  - The tactus is considered as the most salient hierarchical level, consistent with the notated meter, or the foot tapping rate (Desain & Honing 1994).

Mental schemas for Meter
- Metrical Profiles (Palmer & Krumhansl 1990)
  - Pre-established mental frameworks (“schemas”) for musical meter are used during listening.

Syncopation
- Listener judgements of musical complexity are correlated with degree of syncopation (i.e. note location within the beat) (Shmulevich & Povel 2000, Smith & Honing 2006).
  - Compared judgements against formal model of syncopation (Longuet-Higgins & Lee 1984).

Active Rhythm Perception
- Viewed as a resonance between top down and bottom–up processes (see e.g Desain & Honing 2001):

Onset–detection vs. Beat–detection
- Traditionally beat detection relied on accurate onset detection.
  - i.e from MIDI data for Score Following (Dannenberg 1991, Cont 2009).
- This can be difficult for MIR from polyphonic audio recordings.
  - A higher freq. Onset Detection Function from the entire audio signal can be used for beat tracking without all onsets being detected (Schloss 1985, Goto & Muraoka 1994, Scheirer 1998).
The Onset Detection Function

- Represents:
  - Ideal: Each note that contributes to the beat.
  - Practice: Combined envelopes of all notes.
- Tends to emphasise:
  - strong transients (i.e. impulsive sounds)
  - loud notes
  - bass notes
  - wide-band spectrum events (e.g. snare drums).

Dixon’s Envelope Onset Detection

Example Onset Detection

- Pre-processing
- Filtering
- Down-sampling
- Difference function

Common ODF methods

- e.g. (Bello et. al 2005, Dixon 2007, Peeters 2007)
- Optional pre-rectification filtering.
- Envelope mixture from rectification/energy.
- Smoothing of envelope (LP filter).
- Down-sampling for data reduction.
- \( \frac{d}{dt} \log E \) highlights perceived impulses.
- Weighting higher frequencies captures wide-band events.
- Spectral difference between STFT frames.

Existing Rhythmic Models

- Parsing metrical grammars (Longuet-Higgins and Lee 1982).
- Forward projection of likelihood (Desain 1992).

Approaches to beat tracking considered

- Autocorrelation
  - Finding Periodicity in the ODF.
- Beat Spectrum approaches:
  - Spectrum of the ODF.
  - Multi-resolution representation of ODF.
- Dynamic Programming approaches.
  - Efficient selection of correct beat interval.
**Resonator outputs of ODF modulation will reverb** of the ODF. Filterbanks of tuned resonators (i.e. “rhythmic reverb”) of the ODF. Resonator whose resonant F matches rate of ODF modulation will phase-lock. Resonator outputs of common freq summed across subbands:

\[ T = \arg \max_s \sum_s F_{rs} \]

**Beat spectrum methods** (Scheirer 1998)

- Filterbanks of tuned resonators (i.e. “rhythmic reverb”) of the ODF.
- Resonator whose resonant F matches rate of ODF modulation will phase-lock.
- Resonator outputs of common freq summed across subbands:

**Beat Tracking by Peeters (2007)**

- Onset–energy function
- Log–win
- Threshold = 15 dB
- Low-pass filter
- High-pass filter (400 Hz)
- Half-wave rectification
- Anti-alias filtering
- Hamming window

**Multiresolution**

- Auditory–Motor “Primal Sketch” from Sombrero filter banks (Todd 1994, Todd, O’Boyle & Lee 1999)
- Continuous wavelet transform of rhythmic signals (Smith 1996, Smith & Hering 2008)
**Wavelet time–frequency analysis**

Continuous wavelet transform (CWT) decomposes (invertibly) a signal onto scaled and translated instances of a finite time “mother function” or “basis”.

\[
W_s(b, a) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} s(\tau) \left( g \left( \frac{\tau - b}{a} \right) \right) d\tau, \quad a > 0
\]

\[
g(t) = e^{-t^2/2} \cdot e^{i\omega_0 t}
\]

**Wavelets for Rhythm** (Smith & Honing 2008)

- The CWT enables representation of temporal structure in terms of time varying rhythmic frequencies.
- Produces magnitude and phase measures which reveal time–frequency ridges indicating the frequencies present in the input rhythm signal (collectively a skeleton, Tchamitchian & Torrésani '92).

**Implementation**

- Implemented as a set of complex value bandpass filters in Fourier domain.
- Scaling produces a “zooming” time window for each frequency “scale”.
- Creates simultaneous time and frequency localisation close to the Heisenberg inequality.

**Memory Based Tactus**

Wavelet rhythm analysis is also applicable to continuous onset salience traces from auditory models (Coath, et. al 2009).

**Foot–tapping to singing**

- Uses lossy windowed integrator to amass tactus likelihood.
- Suppress all but the magnitude coefficients of the extracted tactus ridge.
- Invert the extracted tactus ridge and original phase plane back to the time domain.
- Creates a single beat oscillation.
- Nominating a starting beat and noting its phase, all other foot–taps are generated for the same phase value.

- Singing examples of Dutch folk songs from the "Onder de Groene Linde" collection (Meertens Institute) using memory based derivation of tactus:
  - Example 1: Original... + Accompaniment.
  - Example 2: ...Original + Accompaniment.
Dynamic Programming (Ellis 2007)

- Goal to generate beat times that match onsets and have near constant IBI.
  \[ C(t_i) = \sum_{i=1}^{N} O(t_i) + \alpha \sum_{i=2}^{N} F(t_i - t_{i-1}, \tau_p). \]
- \( F(\Delta t, \tau) = -\log(\text{actual IBI/ideal IBI})^2. \)
- Ideal IBI from tempo estimation from weighted autocorrelation.
- Recursively calculates max \( C^*(t) \) starting from \( t=0-2\tau \), finding times of max(\( F + C^*(\tau) \)).
- Chooses final max \( C^*(t) \) from last interval, backtraces the saved times.

Beat Histograms

- Summarises rhythmic behaviour of a piece for similarity measures, classification etc.
- Pampalk, Dixon & Widmer (2003)
  - Uses summation of comb filters of Scheier, not just argmax, for comparison.
  - Tempo histogram is weighted using a preference model (van Noorden & Moelants 1999).
  - PCA used to reduce 2000+ 60 dimensions for matching.

Beat Histograms (Tzanetakis and Cook, 2002)

- Similar approach using Autocorrelation.
- Add the amplitudes of the top 3 AC peaks to histogram at each frame.
- Beat histograms are reducible to single features including sum and peak/mean.

Fluctuation Patterns

- Also summarises rhythmic behaviour.
- FFT of envelope: the fluctuation (AM) frequency of the perceived loudness of critical bands (log spectral) (represented on the Bark scale).
- 20 Bark x 60 BF matrix \( \rightarrow \) PCA for matching

```
<table>
<thead>
<tr>
<th>Rock</th>
<th>DJ</th>
<th>11.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>In</td>
<td>Stereo</td>
<td>9.9</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>3.1</td>
<td>0.2</td>
<td></td>
</tr>
</tbody>
</table>
```

Median of the fluctuation patterns of examples of (L-R) Heavy Metal, Dance and Pop. Y axis shows critical bands (Bark 1-20). X axis shows beat frequencies 0-10Hz (0-600BPM). From Pampalk, Rauber & Merkl, (2002)

Meter estimation

- Requires measure (“bar”) period and phase (downbeat) identification.
- Measure period reasonably successful, albeit with octave errors.
- Downbeat identification much harder!
- Genre dependent.

Joint estimation of chord change and downbeat (Papadopoulos & Peeters 2008)

- Hidden Markov Model:
  - States: 24 Major & Minor triads \( \times 4 \) positions within the Measure (pim) for (4/4 time signature).
  - Computes chroma features at each beat.
  - Assumes independence between beat position and chord type: \( P(O|s) = P(O|c) P(O|pim) \)
  - Transition probabilities enforce sequential beats & likelihood of chord transitions.
- Optimal state determined by Viterbi decoding.
  - Chord progression detection improved using metrical knowledge.
  - Identification of downbeats aided by harmonic information.
Review

- Modeling rhythm requires representing perception
- Onset detection functions capture significant events
- Multiple approaches to beat-tracking represent competing perceptual models
- Beat-tracking enables higher-level rhythmic features (FP, BH)
- Beat-tracking enables multi-modal estimation (e.g., down-beat)

Applications

- Low-hanging fruit
  - Basic non-real-time feature extraction
  - Bulk feature extraction into a DB
  - Real-time feature extraction and mapping to synthesis or control
  - Song clustering based on feature vector similarity, clustering, ...
  - PCA of feature spaces using Weka
  - Segmentation based on inter-frame distances

APIs for MIR Tools

- Marsyas: G. Tzanetakis (11), flexible tool set, scripting language, segmentation and classification
- LibOFA: Holm/Pope (00), simple FV for unique ID comparing to a large pre-analyzed database
- D2K/M2K: West/MIREX (06), Java-based GUI related to D2K, many apps.
- LibTSP: P. Kabal (00), C routines for DASP & IO
- CSL: STP/MAT (05), C++ class library for DASP, synthesis, control, spatialization and MIR

APIs – 2

- Libxtract
- Aubio
- SonicVisualizer plug-ins
- Loris
- SPEAR
- CSL
- LibTSP

Spectral Tools

- SPEAR
- Loris
- Marsyas
- Sonic visualizer

Code Exercises

- Buffer, Window classes (see CSL)
- Analyzer class (Marsyas)
- Driver, main(), aubio, libxtract
- IO libraries (libSndFile, PortAudio)
- DASP libraries (libTSP, etc.)
- Starter apps: simple analyzer, sing-along
Q&A

Lab 2

- Feature extraction and flexible feature vectors in MATLAB, Marsyas, Aubio, libExtract
- MATLAB/Weka code for sound clustering with a flexible feature vector
- C++ API examples Marsyas, Aubio, libExtract – pre-built examples to read and customize
- Goal: extract CAL 500 per-song features to .mat or .csv using features from today.

Example Code 1

- AFsp-v9r0 - General-purpose audio file code in C, Peter Kabal @ McGill
- aubio-0.3.2 - library for audio labeling, P. M. Brossier and J. P. Bello, http://aubio.piem.org
- beatDetect - MAT 2450C project by Philip Popp (Xcode)
- bp_proj - Neural Net demo for VisualStudio
- CNMAT-SDIF-alpha - Spectral Data Interchange Format code from UCBerkeley
- dance-o-matic - MAT 240F project by Philip Popp (Xcode)
- EricNewman - Various projects including MAGIC from Eric Newman @ UCSC (Xcode)
- fann-0.2.0.0 - Fast Artificial Neural Network Library, http://leenissen.dk/fann
- FFTW - Fastest Fourier Transform in the West, FFTW.org
- FlowDesigner-0.8.0 - Flow Designer, like SimuLink, jean-marc.valin@usherbrooke.ca
- FlowDesigner-0.9.1-Darwin.pkg - Mac installer into /usr/local/include, etc.
- getRMS2 - store the windowed RMS values of a given input file into a given output file (Xcode)

Example Code 2

- ICA - Independent component analysis code, Shiro Ikeda, shiro@ikeda.cc
- JMARF - "MODULARIZED AUDIO RECOGNITION FRAMEWORK" Serguei Mokhov, The MARF Research and Development Group, Montreal
- libneural-1.0.3 - simple Back-propagation Neural Network, Daniel Franklin
- libofa - Open FingerPrint Architecture, S. T. Pope & Frode Holm, MusicIP (RIP)
- libsndfile - awesome sound file API from Erik de Castro Lopo <erikd@mega-nerd.com>
- libtsp-v7r0 - General-purpose DASP code in C, Peter Kabal @ McGill
- libextract-0.6.3 - library of audio feature extraction functions by Jamie Bullock
- m2k - Music-to-Knowledge in Java (stale?), Kris West, kw@cmp.uea.ac.uk, http://www.music-ir.org
- marf0/2 - "MODULARIZED AUDIO RECOGNITION FRAMEWORK"
- marsyas-0.4.3 - MARSYAS C++ library for MIR, George Tzanetakis
- moc-0.1.1 - "Master of Celebration" playlist generator by Dominik 'Aeneas' Schnitzer

Example Code 3

- SampleAnalyzer - MAT 240F example code, reads sample files and runs analyzers
- sing_along - MAT 240F example code by STP, play a sine wave along with a singer
- sndan - SNDAN, James Beauchamp, implementation of MQ tracking and analysis
- sonic-visualiser-1.8 - program for viewing and analysing music files
- Sonic Visualiser-1.8.dmg - Mac binary installer
- SpectralTracker - MAT 240F example code by Matthew Crossley
- SPEAR_latest.dmg - "Sinusoidal Partial Editing Analysis and Resynthesis", Michael Klingeill
- sphinx3-0.6 - CMU Sphinx Speech Recognition tools
- SPRACHcore-2004-08-26 - Connectionist speech recognition software by Dan Ellis
- STFT - Lance Putnam’s C++ wrapper object for FFTW
- svlib - C++ class library for automatic speech recognition and speaker recognition, jialong.He@bigfoot.com
- tap_alongPP - MAT 240F example code by S T Pope, play a sine wave along with a singer
- ww_beat_tracker.c - Will Wolcott’s simple beat tracker from MAT 240F

Lab 2 – Where to start

- Running C/C++ Examples
  - Using the UNIX shell
  - Using Makefiles
    - apt-get, tar xvf, cd,
    - ./configure --help,
    - ./configure, make, sudo make install
  - Using C/C++ IDEs
    - Eclipse, XCode, VisualStudio
    - Code editing
    - Project mgmnt
    - Debugger
Lab 2 – Where to start

• The Hell that is C/C++ Development
  – UNIX packages and configure scripts
    • Fixing broken configure scripts
  – “Make” packages: make, gmake, cmake
    • Fixing broken makefiles
  – Compiling: getting the right package includes
    • Versions of C, of the std headers
  – Linking: finding the (static & dynamic) libraries
    • Linux vs MacOS or MS–Windows
  – The (truly sad) good advice: minimize the number of libraries you use (JUCE + FFTW)

• Debugging C/C++
  – Anti–bugging techniques
  – Print statements
  – Breakpoints

• Problems
  – Compile–time (includes)
  – Link–time (libraries, modules)
  – Run–time
    • Initialization errors
    • Malloc/free new/delete, garbage collection
    • Logic errors

Lab 2 – DASP Coding

• Support libraries – I/O, DB, ...
  – LibSndFile
  – RTaudio/RTmidi
  – AFsp
  – JUCE
  – FFTW
  – DB APIs: MySQL, PostgreSQL, XMP, JSON ...

• General–purpose DASP Libraries
  – LibTSP, CSL, others
  – Handling main(), set-up/clean–up and data I/O

• Aubio
  – configure, make
  – audioquiet.c
  – audioonset.c
  – SWIG interfaces

• Libxtract
  – simpletest.c – spectrum extraction
  – Max/Pd plug–ins

• Marsyas
  – Setting up & using Cmake
  – sfinfo app
  – pitchextract app
  – bextract app

MIR Code Examples (in C/C++)

• Aubio
  – configure, make
  – audioquiet.c
  – audioonset.c
  – SWIG interfaces

• Libxtract
  – simpletest.c – spectrum extraction
  – Max/Pd plug–ins

• Marsyas
  – Setting up & using Cmake
  – sfinfo app
  – pitchextract app
  – bextract app

MIR Code Examples

• MAT240F examples & student code
  – SampleAnalyzer (fftw, libsndfile)
    • Single–file or file–list processing
    • DB, noDB
  – getRMS2 (libsndfile, libtsp)
    • Batch analysis
    • Windowed RMS, 2nd–stage autocorrelation
  – sing_along (portaudio, fftw, libtsp)
    • Several pitch detectors
    • Runs processing in portaudio call–back
  – Extensions...