8 Audio Retrieval

8.1 Statistical Features

- Typical features:
  - Loudness, Bandwidth, Brightness, Zero-Crossing Rate, Pitch, …
  - Statistical values efficiently describe audio files by feature vectors
  - Do not consider details such as rhythm and melody
  - Only query by example search makes sense

8.1 Music Retrieval

- In order to effectively differentiate complex pieces of music other information are required
- The melody is usually of central importance
  - Simple melody line
  - Formulation of the query, not only by sample audio files:
    - Query by humming
    - Query by whistling
    - Query by singing
    - …
8.1 Music Retrieval

- To establish the melody we first need to detect the notes from the audio signal
  - Many (often overlaid) instruments (possibly each with slightly different melody)
  - Singing

8.1 Query by Humming

- Model based approach
  - Only the characteristic melody should be used for music retrieval, and not the whole song
- What is the melody?
- How to represent melodies?
- How do typical queries look like?

8.1 Model based approach

- Steps:
  - Formulation of the query by humming, whistling, singing or an audio file
  - Extraction of the melody from the recording (spectral analysis, noise reduction, pitch tracking, ...)
  - Encoding the melody (Parsons code, differential code, ...)
  - Comparison with the database
  - Return of the results

8.1 Architecture

8.1 Input

- Singing
  - Difficult because of different talents and strong individuality
- Humming
  - Original idea (Ghia and others, 1995)
  - Often with sound “ta” for note separation
- Whistling
  - Little individuality and good note separation

- Input by virtual instrument (e.g., Greenstone library, New Zealand)
8.1 Conversion

- **Digital Recording**
  - Low sampling rate is sufficient
  - Noise reduction is often necessary
- **Grouping of samples in overlapping frames**
  - Frame size: approximately 50 milliseconds
  - Frame structure: each half-overlapping with the previous one and the subsequent frame
  - Ignore the first frame (start noise)

- For each frame, its **spectral sample** is calculated (short-time FFT)
- Calculation of **pitch** per spectrum, with average amplitude as volume
- If the volume of the pitch is too low, or it cannot be determined, mark the frame as a “silent” frame

8.1 Conversion

- **Find the note boundaries**
  - Boundaries of notes are marked by silent frames or sudden frequency jumps
  - At frequency jumps or sharp jumps in the volume
    - Add a new silent frame
  - The ratio of successive frequencies exceeds a threshold (about 3%):
    - Add a new silent frame

- Between two frames with the same frequency, there is a single frame with a different frequency:
  - **Smoothing**: replace the deviating frequency by the frequency of the neighbors
  - Between two silent frames, the frequency varies only slightly:
    - Replace the frequencies by the average frequency

8.1 Conversion

- **Connecting the same pitches to notes**
  - Connect all pitches between silent frames into a note with a higher duration (depending on the number of frames)
  - Remove notes below a specified minimum length
  - Remove all silent frames
- **Output**: melody with note height and duration

- Melody only needs to be **sufficiently well** represented
  - More accurate representation leads to larger amounts of data
  - Accurate representation by MIDI data (height, start time, duration, …)
- Simpler systems use only a rough classification of the melody

8.2 Symbolic Representation
8.2 The Parsons Code

- Simple classification of melody lines (Parsons, 1975)
- Sequence of note variations (Chain code)
  - U (up) at higher note
  - R (repeat) at the same note
  - D (down) at a lower note
- The first note is used just as reference (symbol: ◊)

8.2 The Parsons Code

- Example: Beethoven’s “Ode to Joy”
  - RUURDDDDRUURDR ...

8.2 The Parsons Code

- Ignores characteristics such as rhythm or precise note intervals
- An advantage is the high fault tolerance, especially towards the query
  - Input in third, fourth, … octave
  - Inadequate rhythm
  - Regardless of scale: major or minor (no transposition required)

8.2 The Parsons Code

- Parsons “The Directory of Tunes and Musical Themes”

Some examples:
- “Paula’s Theme” (Pauline’s Theme)
- “Black Night” (Black Night)
- “Love Me Tender” (Lover Me Tender)
- “White Christmas” (White Christmas)
- First verse in “My Way” (My Way)
- First verse in “The Sound of Music” (The Sound of Music)
- Verse in bitter line (Bitter Line)
- First verse in “Shadows in the Night” (Shadows in the Night)

8.2 Parsons Code Matching

- Parsons code of the query, has to be compared with all the codes from the database
- Matching using the edit distance
- Since we do not know at which point the query melody fragment occurs, matching must be performed on substrings

8.2 Parsons Code Matching

- Typical errors in music editing distance
  - A note is left out
  - One false note is added
  - An existing note is sung wrong
  - Several short notes are combined
  - Long notes are fragmented
8.2 Parsons Code Matching

- Given: two strings
  \( A = a_1, a_2, \ldots, a_m \) and
  \( B = b_1, b_2, \ldots, b_n \)

- Goal: intuitive measure of the dissimilarity \( d(A, B) \) of strings \( A \) and \( B \)

- Again edit distance:
  Convert \( A \) to \( B \) by using fixed operations; find the sequence of operations with minimum cost

  - Why not compare note by note?

8.2 Parsons Code Matching

- Operations:
  - Delete a single character
  - Inserting a single character
  - Replacing a single character
  - Replace a string of characters by a single character (consolidation)
  - Replacing a single character by a sequence of characters (fragmentation)

- Every character of \( A \) and \( B \) must be involved in exactly one operation!

- Cost table: entry \( w(x, y) > 0 \) indicates the cost of the replacement of a string \( x \) by \( y \)

8.2 Parsons Code Matching

- Example:
  \( A = RUDRR \)
  \( B = UUDR \)

- If we apply (as in the example), the operations from left to right, then every node results from a prefix of \( B \) and a suffix of \( A \)
  - E.g., \( A = RUDRR, B = UUDR \)

- This means a total of \( O(m \cdot n) \) vertices in the graph, which is much less than the number of all paths from \( A \) to \( B \)

8.2 Parsons Code Matching

- Since all costs are non-negative, one can find a path from \( A \) to \( B \) with minimal cost simply by means of dynamic programming (Mongeau and Sankoff, 1990)

- Examples:
8.2 Parsons Code Matching

• Cost values need to be adapted to typical input errors, of the user
  – Replacing R...R with R is an usual error
  – But: UD and DU is not, and the cost should be higher
  – Inserting R should therefore be cheaper than insertion of U or D (about half the cost)
  – At replacements, the costs of operations R→D and R→U should be smaller than that of D→U and U→D

8.2 Parsons Code Matching

• Dynamic programming considers all possible nodes in the graph, no matter how large the associated edit costs are
• But the query should not differ too much from the result
  – Therefore: ignore nodes in the graph:
    • Which are only reachable from A at high cost
    or
    • From which B can be reached only at high cost
  – To do this we can use a matrix of nodes of A and B

8.2 Parsons Code Matching

• Melody search is matching substrings: song A is longer than query B, but B can start at any point of A
  – The database should know where to look
  – Otherwise costly

8.2 Difference Codes

• Parsons code ignores the strength of the pitch change
• Difference codes save these interval information as number of semitones on the MIDI scale (12-tone scale)
8.2 Difference Codes

- Beethoven’s “Ode to Joy”
- Parsons Code:
  ◊ R U U R D D D D
  Difference Code:
  ◊ 0 1 2 0 −2 −1 −2 −2
- But also bigger jumps:

8.2 Difference Codes

- Distribution of intervals in a music database with about 10000 songs (Kosugi and others, 2000)

8.2 Difference Codes

- Advantages
  - Allows precise distinction of music, by considering also the size of the jump interval in the weighting of the edit distance
- Disadvantages
  - It also requires more effort in matching and a more accurate note segmentation
  - The result is very dependent on the audio collection, since melodies often have little tone steps and also on the users

8.2 Frame based Representation

- Precise segmentation of the query and the music in the database is essential for both the Parsons code, and the difference code
- Frame based representations do not segment notes, but only use the contour of the melody

8.2 Frame based Representation

- Frame classification should be equidistant
  - Not a frame of 10 ms and one of 100 ms
- Advantages:
  - No inaccuracies by incorrect segmentation
  - Frame sequences also contain the rhythm information
- ... But the retrieval time is also significantly higher
8.2 Frame based Representation

- The frame-based representation leads to a time series of pitch values
- **Point wise comparison** of the sound contour leads to very poor results because:
  - Speed of query might be different from the speed of objects in the database
  - The rhythm in the query is often wrong

8.2 Dynamic Time Warping

- Dynamic matching between contours is required (the “singing length” of the notes plays a minor role)
- Known method from the Data Mining: **Dynamic Time Warping** (DTW; Berndt and Clifford, 1994)
  - Distance measure for time series
  - Same principle as edit distance
  - The only difference: no finite alphabet (e.g., U D R in Parsons code) anymore, but continuous numbers
  - The cost of an operation depends on the values of the involved numbers

8.2 Illustration

Paths on a two-dimensional map of time (0, 0) to (M, N) are valid matching

8.2 Warping Paths

- Monotony: \( i(k) \leq i(k+1) \) and \( j(k) \leq j(k+1) \)
- Continuity: \( i(k+1)-i(k) \leq 1 \) and \( j(k+1)-j(k) \leq 1 \)
- Boundaries: \( i(1) = j(1) = 1 \), \( i(K) = N \) and \( j(K) = M \)
- Calculation using dynamic programming in \( O(m \cdot n) \) time
- In special cases even faster...

8.2 DTW

- DTW example
8.2 Uniform Time Warping

- The uniform time-warping distance between two time series \( x \) and \( y \) is defined as:
  \[
  D_{UTW}(x, y) = \sum_{i=1}^{\min(m,n)} |x_i - y_i|^2 / \min(m,n)
  \]
- Both time-axis are extended to \( mn \) (or to the least common multiple of \( m \) and \( n \))
  - Problematic for time series with variable speed

8.2 Local Dynamic Time Warping

- Intuitive matching for humans
  - Extend both series to the same length
  - Compare pointwise, but allow little warping intervals
- So again: extend the calculation on one area, which lies near the matrix diagonal

8.2 Local Dynamic Time Warping

- With LDTW distances we can build effective indexes for comparing time series (in our case, melodies)
- Extension of the GEMINI approach by envelopes (Zhu and Shasha, 2003)
  - Calculate the envelope for a query and cut with high-dimensional index structure

8.2 Example

- Idea of the uniform time warping is that warping paths should be as diagonal as possible
  - But UTW can also be calculated from time series of different lengths
- Uniform time warping is a generalization of the Time Scaling

- Example:
  - Time complexity of LDTW is \( O(kn) \) where \( k \) is the width of the strip

[Zhu and Shasha, 2003]
8.2 Example

(Zhu and Shasha, 2003)

• After transformation into special normal forms:

8.3 Example

• Acoustic events
  – Frame based methods shows the behavior of the audio signal, but we don’t know what this behavior means
  – How to determine acoustic events in the audio signal?
    • What has caused this particular signal path? (e.g., could it be the beginning of a note?)
    • More or less plausible explanations
    • How can we model it?

8.3 Example

• The observation could either be:
  – Independent short note on semitone 53
  – Or only envelope attack signal for semitone 52 (at sustain level)

8.3 Example

• Implementation of the (hidden) sequence of events in a string (over a fixed alphabet)

8.3 Example

• Acoustic event “single note” as a sequence of “atomic events” according to the envelope model
  – State set $Q = \{A, D, S, R, \epsilon\}$
    • These states represent attack, decay, sustain, release and silence
  – Possible state transitions are determined by a Markov chain (stochastic variant of finite automata)

8.3 Example

• Homogeneous Markov process:
  – In each state the outgoing edge weights add up to 1
  – Transition probabilities are time-invariant
  – An start distribution is defined
8.3 Start Distribution

- Start distribution for each node determines the probability that the process starts in this node.
- Example: single note always starts with attack
  \[ \pi : Q \to [0, 1] \]
  with \( \pi(A) = 1 \) and \( \pi(D) = \pi(S) = \pi(\varepsilon) = 0 \)

8.3 Example

- Appearance probability subsequently ADSSR\(\varepsilon\):
  \[ 1 \cdot 0.3 \cdot 0.6 \cdot 0.7 \cdot 0.3 \cdot 0.5 = 0.0189 \]
- Appearance probability subsequently ADDDSR\(\varepsilon\):
  \[ 1 \cdot 0.3 \cdot 0.4 \cdot 0.4 \cdot 0.6 \cdot 0.3 \cdot 0.5 = 0.0043 \]

8.3 Basic Problem

- Detection of acoustic events (such as single notes) is from the audio signal, almost impossible
- Solution:
  - State sequences detection must also be probabilistic
  - “If the signal has the observed shape, then I am very likely in state x or less likely in state y”

8.3 Observations

- Finite class of possible observations
  - E.g., \( O = \{o_1, \ldots, o_6\} \)
- The probabilities that observation \( o_i \) is made in state \( q \in Q \), are required
  \[ p_Q : O \to [0, 1] \]
  - E.g., \( p_{0A} \cdot 0.7 \)

8.3 Overall Probability

- Observation “\( o_3 \cdot o_5 \cdot o_1 \)”
- How high is the probability that the model “ADS” was responsible for this observation?
  \[ \pi(A) \cdot p_A(o_3) \cdot p_{A,D}(o_5) \cdot p_{D,S}(o_1) \]
- “ADS” is just a supposition
  - The “true” model is hidden (thus: Hidden Markov Model)

8.3 The Real Problem

- Known sequence of \( n \) observations
  \( o_{i_1}, \ldots, o_{i_n} \in O^n \)
  - What is the most likely state sequence \( q_{i_1}, \ldots, q_{i_n} \in Q \)?
  - Is it possible to assign the sequence of observations, an overall probability of the event “single note”? (with respect to the specific model \( Q \))
We can assign a sequence of observations to the acoustic event, whose HMM has created the observations with the **highest** probability.

### 8.3 Acoustic Events

- **Observations**
- **Hidden States**

### 8.3 Conditional Probabilities

- Probability of event A if it is already known that event B has occurred:

\[ P(A|B) := \frac{P(A \cap B)}{P(B)} \]

- Analogously for the probability densities of random variables X and Y:

\[ P(X|Y) := \frac{P(X,Y)}{P(Y)} = \frac{P_X Y}{P_Y} \]

### 8.3 Stochastic Processes

- A **stochastic process** is a sequence of random variables \((X_0, X_1, X_2, \ldots)\).

- A **Markov process** additionally satisfies the Markov condition:

\[ \forall n \in \mathbb{N} \text{ and } i_0, \ldots, i_{n+1} \in I \text{ mit } P(X_0 = i_0, \ldots, X_n = i_n) > 0 \]

\[ P(X_{n+1} = i_{n+1}|X_0 = i_0, \ldots, X_n = i_n) = P(X_{n+1} = i_{n+1}|X_n = i_n) \]

- Remember Markov property by textures (neighborhood)!

### 8.3 Stochastic Processes

- Markov processes are **homogeneous** if the transition probability \(p_{ij}\) from state \(i\) to state \(j\) are independent of \(n\):

\[ \forall n \in \mathbb{N} \text{ and } i, j \in I : P(X_{n+1} = j|X_n = i) = p_{ij} \]

- Knowing the initial distribution

\[ \forall i \in I : \pi_i := P(X_0 = i), \quad \pi := (\pi_i)_{i \in I} \]

we can determine the overall distribution of the process.

### 8.3 Stochastic Processes

- A HMM has at any time additional **time-invariant** observation probabilities

- A **HMM** consists of

  - A homogeneous Markov process \((Q_1)_{i \in \mathbb{N}}\) with state set

\[ Q := \{q_1, \ldots, q_N\} \quad \text{we } \forall t : Q_t : \Omega \to Q \]

  - Transition probabilities

\[ \forall i, j \in [1 : N] : a_{ij} := P(Q_t = q_i|Q_{t-1} = q_j) \]
8.3 Hidden Markov Model

- Start distribution
  \( \forall i \in [1 : N] : \pi_i := P(Q_0 = q_i) \)

- Stochastic process \((O_t)_{t \in \mathbb{N}}\) of observations with basic sets
  \( \Omega := \{o_1, \ldots, o_M\} \) so \( \forall t : O_t : \Omega \rightarrow \Omega \)

- And observation probabilities of observation \(o_k\) in state \(q_j\)
  \( \forall j \in [1 : N], k \in [1 : M] : b_{jk} := P(O_t = o_k | Q_t = q_j) \)

...will be continued next lecture

This Lecture

- Audio Retrieval (continued)
  - Query by Humming
  - Melody: Representation and Matching
    - Parsons-Codes
    - Dynamic Time Warping
  - Hidden Markov Models

Next lecture

- To be continued: Hidden Markov Models
- Introduction to Video Retrieval