Creative Computing II

Christophe Rhodes
c.rhodes@gold.ac.uk

Autumn 2010, Wednesdays:
10:00–12:00: RHB307 & 14:00–16:00: WB316
Winter 2011, Wednesdays:
10:00–12:00: RHB307 & 14:00–16:00: WB316

- Constant-Q spectrum:
  - start with the squared magnitude of the Fourier spectrum bins, but then *combine* into logarithmically-spaced bins;
  - intended to mimic the sensitivity of the basilar membrane;
  - captures the notion of musical pitch;
  - does not capture **octave invariance** (application-dependent whether that is a problem)
- *Chromagram*
  - (usually) starts with a constant-Q spectrum, 12 bins per octave;
  - 'folds' the octaves over: adds values in bins with the same (octave-invariant) pitch;
  - captures the pitch-name content of the audio.
- *Cepstrum*
  - starts (approximately) with a constant-Q spectrum expressed in decibels;
  - take the Fourier transform of that object.
  - captures the idea of the 'timbre' of the sound.

Fourier transform operates on (conceptually) infinite signal

- Transform corresponds to all times at once
- Music / speech / sound not usually a stationary process

Time-localisation: multiply signal by window function

Previously:

- the Fourier Transform of a convolution is the Hadamard product of the individual Fourier Transforms
- used (with the Fast Fourier Transform) in efficient implementation of LTI systems

Also:

- Fourier Transform of Hadamard product is the convolution of the individual Fourier Transforms
- Implies design choices for the window function:
  - Rectangular (Dirichlet)
  - Hann
  - Hamming
  - ...

Multimedia Information Retrieval

Constant-Q spectrum

Short-Time Fourier Transform parameters:

- hop size
- window size

Constant-Q binning parameters:

- low-frequency edge
- high-frequency edge
- bins per octave

The central frequency of bin k is

$$f_k = f_0 \times 2^{\frac{k}{\mathrm{bpo}}}$$

The bandwidth of bin k is approximately

$$f_k \times (2^{\frac{1}{\mathrm{bpo}}} - 1)$$

(so $Q = \frac{1}{2^{\frac{1}{\mathrm{bpo}}} - 1} = 17.3$ for $\mathrm{bpo} = 12$.

Chromagram parameters:

- as STFT and Constant-Q parameters
- bins per octave (almost always) set to 12
- some applications will want 24 or 36

combine / ‘fold’ octave-equivalent bins together
Both the constant-Q and chromagram features can be calculated using Matrix multiplications of the magnitude spectrum.

- start with constant-Q spectrum
- take logs of bin values
- take power spectrum of resulting vector

Represents 'timbre', captures some forms of pitch

- particularly useful in voice / speech recognition
- applications in music retrieval

(More silly names): quefrency, cepstral alanysis, liftering.

The overall architecture:

- a collection of items;
- some set of queries that will be supported;
- some user interface for specifying queries and retrieving results;

Two kinds of error in an MIR system:

- false **positive**;
- false **negative**.

	relevant	irrelevant
retrieved	**TP**	**FP**
rejected	**FN**	**TN**

Two kinds of error in an MIR system:

- false **positive**;
- false **negative**.

	relevant	irrelevant
retrieved	**TP**	**FP**
rejected	**FN**	**TN**

Measures of performance:

- **precision**:
  - what proportion of retrieved results are relevant?
  - $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$

Two kinds of error in an MIR system:

- false **positive**;
- false **negative**.

	relevant	irrelevant
retrieved	**TP**	**FP**
rejected	**FN**	**TN**

Measures of performance:

- **precision**:
  - what proportion of retrieved results are relevant?
  - $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
- **recall**:
  - what proportion of relevant items did I find?
  - $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$

Given an existing database, a query item, and some kind of query specification:

- compute some feature of the query item;
- for each database item
  - compute the same feature for the item;
  - compare the query and database item features with an appropriate distance measure;
  - if the distance is sufficiently small, retrieve the item, otherwise reject it;
- return the collection of retrieved items.

Multimedia Information Retrieval

MIR Systems: Small Collections

Given an existing database, a query item, and some kind of query specification:

- compute some feature of the query item;
- for each database item
  - compute the same feature for the item;
  - compare the query and database item features with an appropriate distance measure;
  - if the distance is sufficiently small, retrieve the item, otherwise reject it;
- return the collection of retrieved items.

Refinements:

- limit the number of retrieved items (by removing larger-distance items from the retrieved set);
- cluster the retrieved set in some way for visualisation;
- infinite possibilities...

The algorithm in the previous slide is $O(N)$ in the size of the database: the feature for each database item is computed and compared.

- precompute database features (easy)

The algorithm in the previous slide is $O(N)$ in the size of the database: the feature for each database item is computed and compared.

- precompute database features (easy)
- precompare features with query (**index**, usually hard)

The algorithm in the previous slide is $O(N)$ in the size of the database: the feature for each database item is computed and compared.

- precompute database features (easy)
- precompare features with query (**index**, usually hard)

For a scalar feature:

- precomparison is in fact easy: binary tree (for example);
- finding the nearest neighbour is $O(\log N)$.

The algorithm in the previous slide is $O(N)$ in the size of the database: the feature for each database item is computed and compared.

- precompute database features (easy)
- precompare features with query (**index**, usually hard)

For a scalar feature:

- precomparison is in fact easy: binary tree (for example);
- finding the nearest neighbour is $O(\log N)$.

For a vector feature:

- precomparison is difficult in general;
- spatial trees, locality-sensitive hashing, probabilistic algorithms;
- (beyond the scope of this course: current research!)