Similarity Searching Techniques in Content-based Audio Retrieval via Hashing

Yi Yu, Masami Takata, and Kazuki Joe

{yuyi, takata, joe}@ics.nara-wu.ac.jp
Graduate School of Humanity and Science
Nara Women's University

Nara Women's University

Outline

- Background and motivation
- Short review -- ANN, LSH and E^{2}LSH
- Proposed framework
- Experiments and results
- Conclusion and future work

Nara Women's University

Content-based Audio Retrieval

- Retrieval based on spectral similarity is difficult
  - High dimensionality of features
  - Complex computation
  - Large database size
- Scalable retrieval capabilities need to be exploited
  - Audio indexing structures
  - Partial sequences comparison

Motivation

- Depend on:
  - Mapping features to integer values by heuristics
  - Reducing pairwise comparisons by hashing
- Challenges:
  - Characterize acoustic objects with relevant spectral features.
  - Represent audio features so that they can be indexed.
  - Locate desired music segments with a given query in acceptable time.

Outline

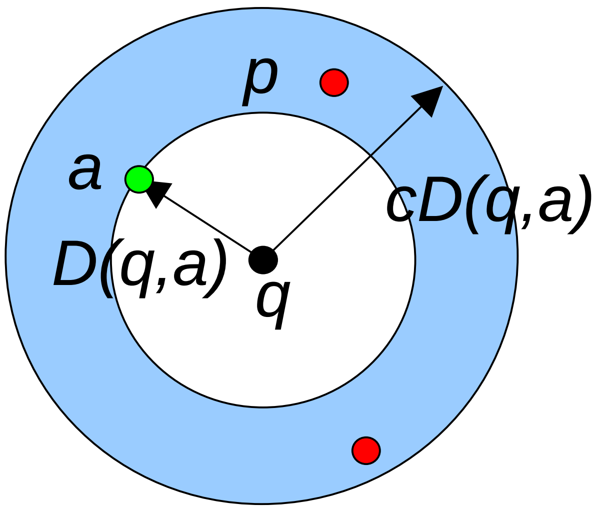
- Introduction and motivation
- Short review -- ANN, LSH and E^{2}LSH
- Proposed framework
- Experiments and results
- Conclusion and future work

Approximate Nearest Neighbor(ANN)

- Given -- a set *P* of *n* points in R^d (*d* - dimension) and a slackness parameter $\varepsilon > 0$
- Goal -- with a query point *q* whose nearest neighbor in *P* is *a*, find one/all points *p* in *P*, satisfying

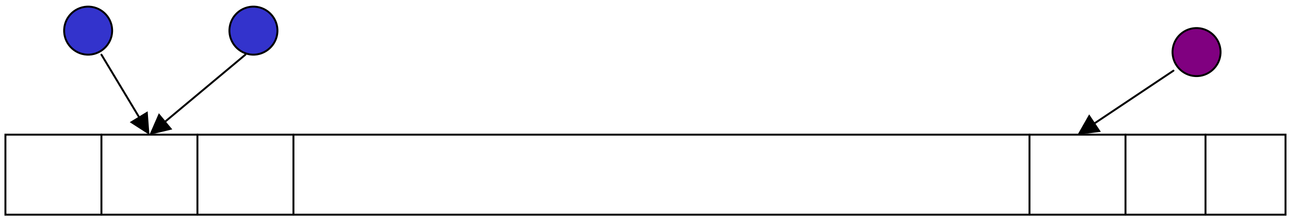
  $$D(p,q) <= c\ D(q,a),\ c=1+\varepsilon$$

- Points in the shadowed ring are desired.



Locality-Sensitive Hashing (LSH)

- Hash function:
  - A pseudo random hash value is obtained
  - Hash value is nearly uniformly distributed.
- LSH: hash function is required to maintain the similarity. For any pair of points *p, q*,
  - Hash function *h*, generate *h(p)*, *h(q)*
  - *Pr[h(p)=h(q)]* is "high" if *p* is "close" to *q*
  - *Pr*[*h*(*p*)=*h*(*q*)] is "low" if *p* is"far" from *q*



Nara Women's University

Exact Euclidian LSH (E^2LSH)

- E^2LSH performs locality-sensitive dimension reduction by p-stable distribution
  - A distribution D over R is called p-stable, if
    (i) for any n real numbers $V = (v_1, v_2, \ldots, v_n)^T$
    (ii) i.i.d. random variables X=($x_1, x_2, \ldots, x_n$) and x with distribution D
    (iii) there exists p, $y = \left(\sum_i |v_i|^p\right)^{1/p} x$ and $f_V(X) = \sum_{i=1}^n v_i x_i$ have the same distribution.
  - Dimension compression $X \rightarrow f_V(X)$

Nara Women's University

Outline

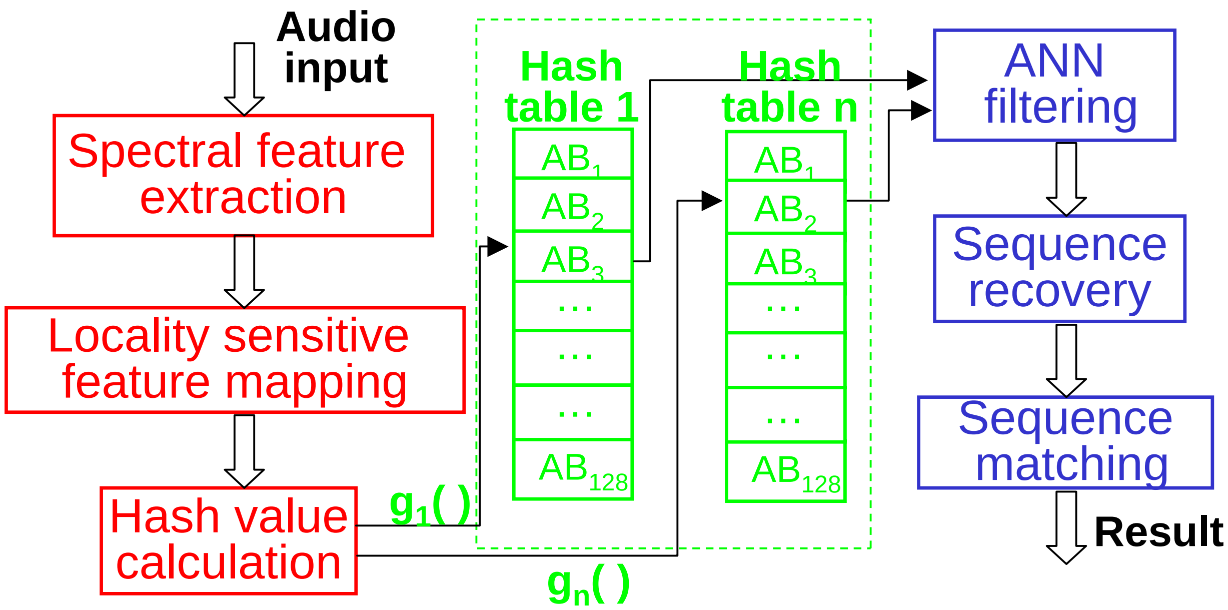
- Introduction and motivation
- Short review -- ANN, LSH and E^2LSH
- Proposed framework
- Experiments and results
- Conclusion and future work

Problem Definition

- Match acoustic sequences without comparing a query to each object in the database.
  - A corpus of n musical reference pieces are represented by frames $R=\{r_{i,j} : r_{i,j} \in R_i, 1 \le i \le n, 1 \le j \le |R_i|\}$
    - $r_{i,j}$ -- j^{th} spectral feature of i^{th} reference melody in a high-dimension space
  - A query sequence $q_1, q_2, \ldots, q_Q$ filters some resemblances by E^{2}LSH/LSH-based ANN.
  - Resembled features are reorganized and compared by DP/Sparse DP.

Retrieval Framework

- Task:
  - Take a fragment of the query song as input
  - Perform a content-based similarity retrieval
  - Return melodies similar to this query fragment
- Major stages:
  - Metadata organization (*red* + *green*)
  - Querying (*red* + *blue*)



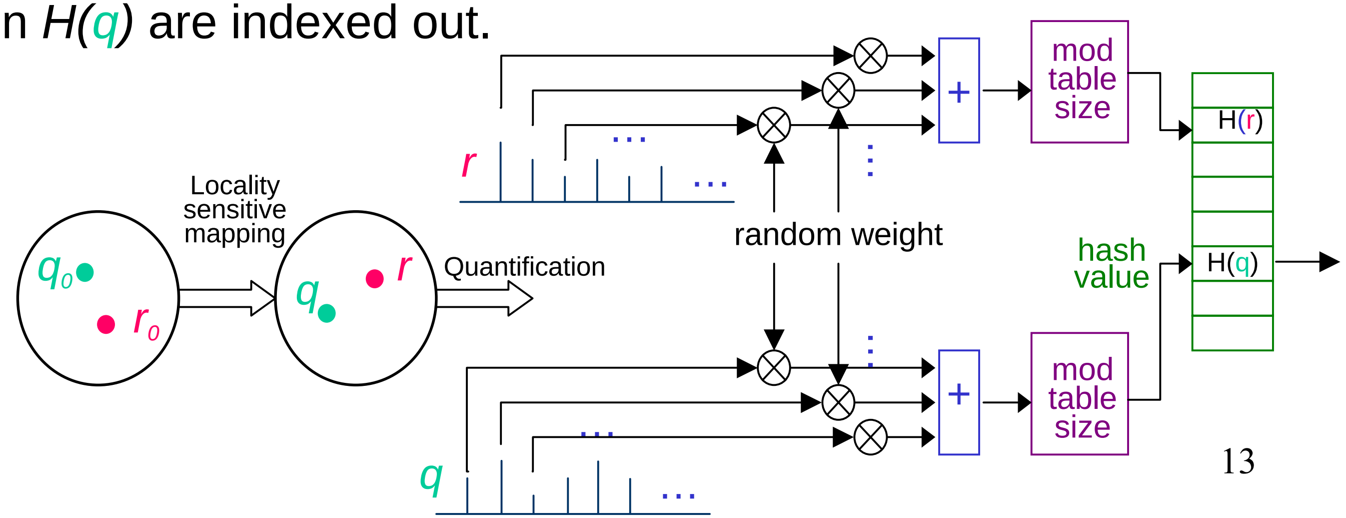
Nara Women's University

Metadata Organization

- Basic procedures:
  - Audio sequences are divided into small frames
    - STFT is calculated and used as the feature
  - Feature mapping and hash value is calculated
    - In LSH (*hash value is directly calculated from STFT*)
    - In E^2LSH (*STFT is first projected to a lower dimensional sub-feature, hash value is calculated*)
  - The features are stored in the bucket
- Results -- Convert audio features into "*indexable*" items.

Example: a Hash Instance

- Original feature (q_0, r_0), Locality sensitive mapping (q, r), Per-dimension quantification, Hash calculation [$H(r)$, $H(q)$]
- Random weight makes hash values of reference melodies almost uniformly distributed.
- If q and r have a short distance
  - They are quantified to same integer sequences
  - & generate same hash value ($H(r) = H(q)$)with a high probability.
  - Features in $H(q)$ are indexed out.



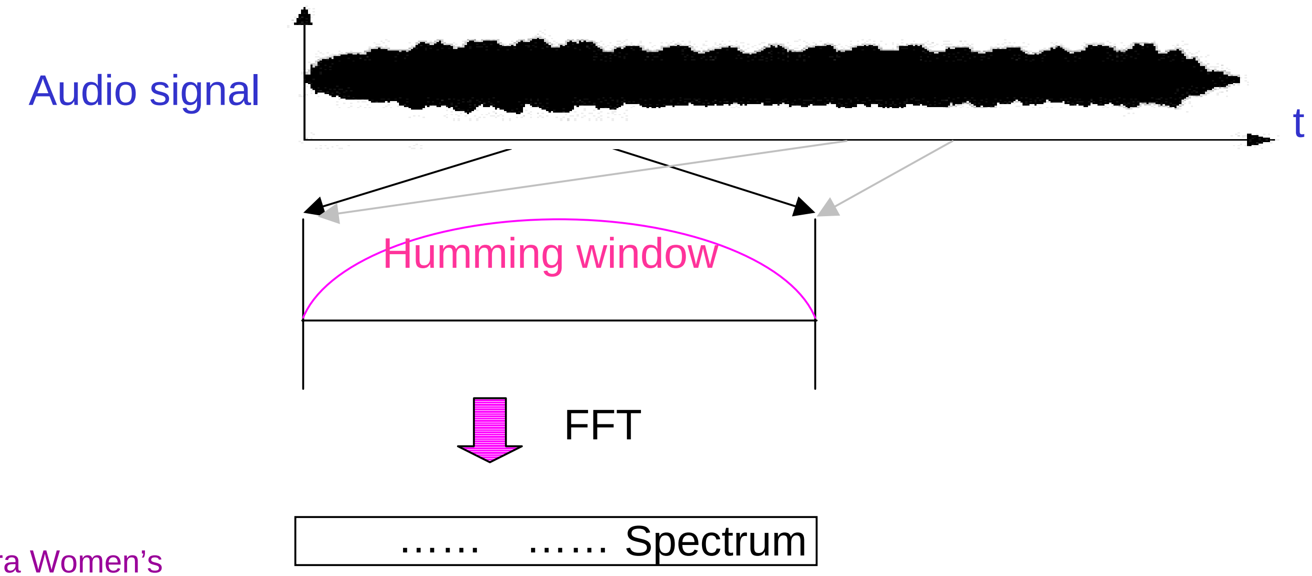
Nara Women's University

Parallel Hash Instances

- **Necessary condition:**
  - Each hash instance contains all the features.
  - Locality sensitive mapping generates different features & keep similarity
- **Parallel lookup:**
  - Construct L hash instances with random $g_1, g_2, \ldots g_L$
  - With a query feature Q, lookup buckets $g_1(Q), g_2(Q)\ldots g_L(Q)$
  - $g_1(Q)\ U\ g_2(Q)\ U \ldots U\ g_L(Q)$ gives total results

Query Stage I

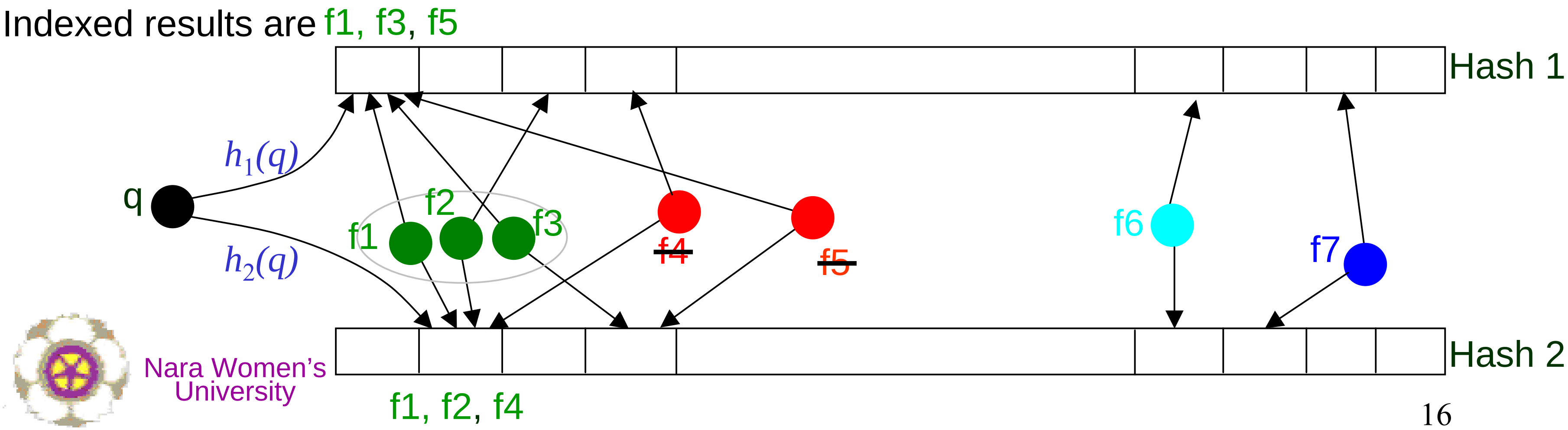
- Feature extraction
  - Divide the query into overlapped frames
  - Calculate STFT for each frame



Nara Women's University

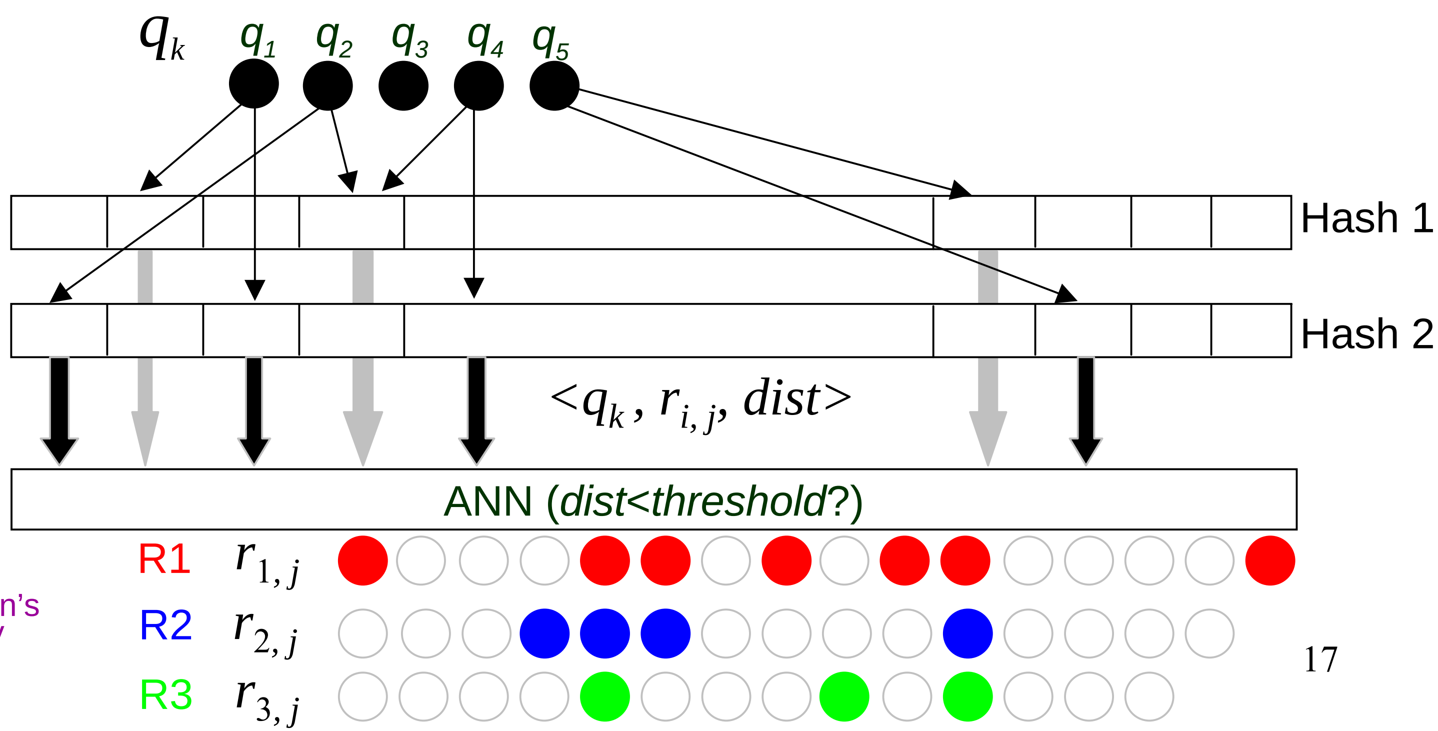
Query Stage II

- Hashing-based ANN:
  - Similar frames lie in the same bucket
  - However, dissimilar frames also exist (~~dissimilar frames~~)
  - Approximation allows a significant speedup of the calculation
- Example(Index with single feature):
  - Assume that q is similar to f1, f2, f3.
    - Lookup hash table 1, $h_1(q)$ gives query result f1, f3 and f5.
    - Lookup hash table 2, $h_2(q)$ gives query result f1, f2 and f4.
    - ~~f5~~ & ~~f4~~ are not similar to q and are removed by ANN.
    - Union of indexed results are f1, f2 and f3.



Query Stage III

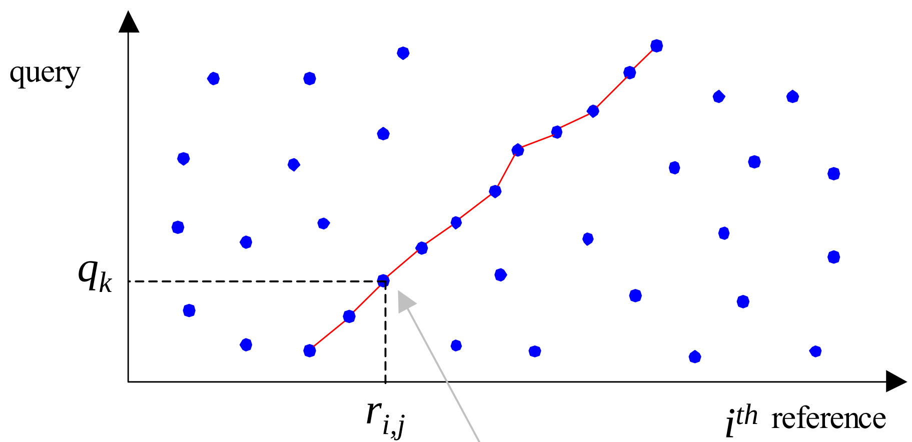
- Find desired target with a sequence of features
  - With query sequences (q_1,q_2,q_3,q_4,q_5) lookup parallel hash tables
    - Matched features belong to 3 reference melodies.
    - They are reorganized in time order.
      7 features in the 1st melody R_1, 4 features in the 2nd melody R_2,
      3 features in the 3rd melody R_3.
    - On this basis, the sequence comparison is performed



Nara Women's University

Query Stage IV

- Matched pairs are sparsely distributed over the Dynamic Time Warping (DTW) table.
  - The conventional Dynamic Programming (DP) is not efficient.
- Our sequence comparison scheme – Sparse DP (SDP)
  - Distance calculated in the filtering stage is converted into weights and filled into the DTW table
  - Melody generating the maximal weight path is the best candidate



Matched pair $<q_k, r_{i,j}, dist>$

weight=1/*dist*

Outline

- Introduction and motivation
- Short review -- ANN, LSH and E^2LSH
- Proposed framework
- Experiments and results
- Conclusion and future work

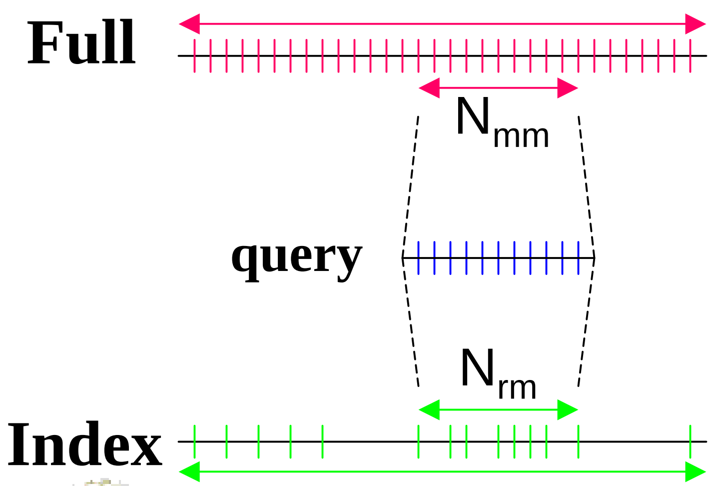
Nara Women's University

Experiment Setup

- System parameters
  - 166 reference melodies, each melody: 60s
  - A query piece: 8s
  - Sampling rate: 22.05KHz
  - Frame length: 1024, Frame overlap: 50%
  - Hash table size: 128
- Experiments goal:
  - Evaluate performance of avoiding full pairwise comparison
  - Compare LSH-DP, LSH-SDP, E^2LSH-DP, E^2LSH-SDP
- Evaluation metric:
  - Matched percentage
  - Computation time
  - Retrieval ratio

Experiments I -- Matched Percentage

- Focus on the accuracy of indexing
  - Ratio N_{rm}/N_{mm} is defined as Valid Match Percentage (VMP).
    - N_{mm}: Frames of the matched part in the desired reference melody under the conventional DP.
    - N_{rm}: Remaining frames of matched part in the desired reference melody after the filtering stage in LSH/E^{2}LSH
  - A good indexing scheme results in a high VMP.



VMP under different filtering threshold (3 hash tables)

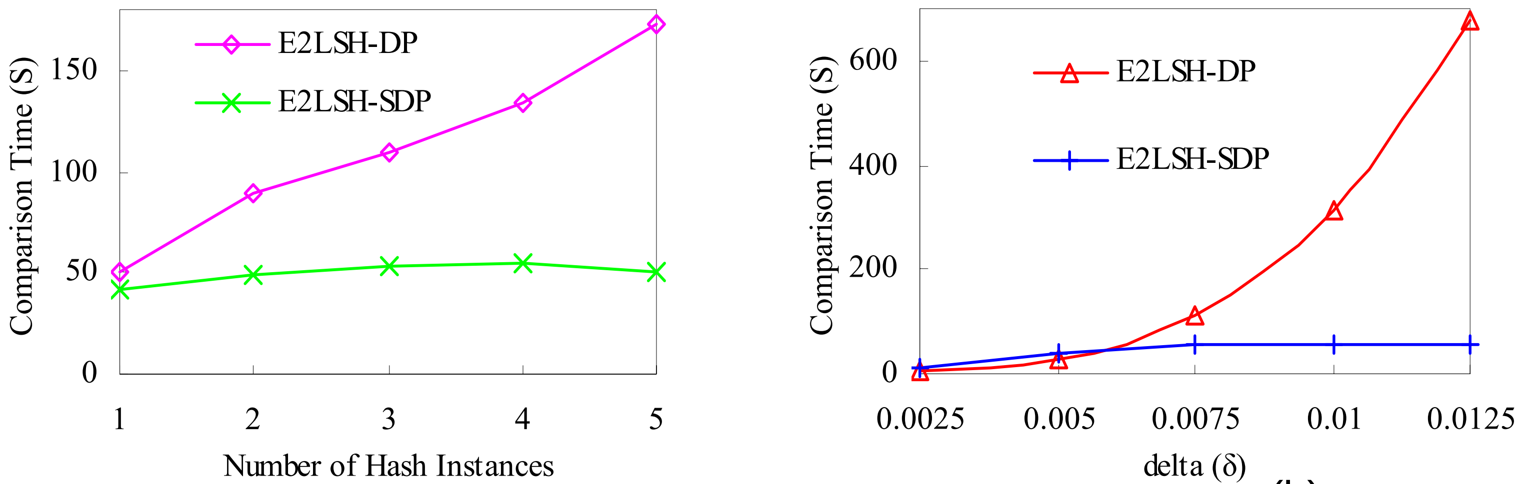
δ_{LSH}	0.01	0.02	0.03	0.04	0.05
VMP_{LSH}	0.133	0.255	0.400	0.537	0.669
δ_{E2LSH}	0.0025	0.005	0.0075	0.0100	0.0125
VMP_{E2LSH}	0.123	0.240	0.363	0.472	0.573

Increasing filtering threshold leads to a high VMP at the cost of more computation.

Nara Women's University

Experiments II -- Computation Time

- Computation is mainly considered in two aspects:
  - *Indexing the features by LSH/E²LSH together with ANN*
  - *Comparing feature sequences*
- Short discussion
  - SDP has a very obvious superiority over DP
    - it avoids the calculation of feature distance
    - & its comparison time approaches a steady value, which guarantees the worst retrieval time.
  - SDP outperforms DP



(a) (b)

Comparison time in DP and SDP under different number of hash tables (δ_{E2LSH}=0.0075) or different filtering threshold δ (3 hash tables)

Nara Women's University

Experiments II -- Computation Time

- All the queries are performed under the different schemes
- Short discussion
  - Conventional DP without hashing takes the longest time
  - E^2LSH-SDP accelerates retrieval speed by 42.7 times compared with conventional DP.

The total retrieval time consumed under different schemes

Scheme	LSH-DP	LSH-SDP	E2LSH-DP	E2LSH-SDP	DP
Time(s)	258.8	213.34	139.5	83.4	3562.2

Experiments III -- Retrieval Ratio

- A tradeoff is made between retrieval ratio and retrieval time
- With a suitable filtering threshold, the retrieval ratio is high enough while the computation time is controlled

Top-4 retrieval ratio of LSH/E^2LSH (3 hash tables) under different filtering threshold δ

δ_{LSH}	0.01	0.02	0.03	0.04	0.05
LSH-DP	088	1	1	1	1
LSH-SDP	0.94	1	1	1	1
δ_{E2LSH}	0.0025	0.005	0.0075	0.01	0.0125
E^2LSH-DP	0.92	0.98	1	1	1
E^2LSH-SDP	0.96	1	1	1	1

δ_{LSH} =0.03 & δ_{E2LSH} =0.0075 are suitable thresholds since a smaller value decreases retrieval ratio while a larger value increases the computation cost.

Outline

- Introduction and motivation
- Short review -- ANN, LSH and E^2LSH
- Proposed framework
- Experiments and results
- Conclusion and future work

Conclusion and Future Work

- Our contribution
  - Established indexed framework for query-by-content audio retrieval
    - Efficiently organizing audio features(E^2LSH/LSH)
    - Efficiently avoiding full pair-wise comparison of audio sequences(SDP/DP)
  - Effectiveness of proposed algorithms(E^2LSH-SDP, E^2LSH-DP,LSH-DP,LSH-DP)
    - Matched Percentage
    - Computation time
    - Retrieval ratio
- Future work
  - Evaluation of scalability of the proposed schemes with a larger database
  - Application of query-by-content audio retrieval in an ubiquitous environment.

Nara Women's University

Thank You!