Authenticated Content-Based Image Retrieval

GUO Shangwei

June 21, 2018

Contents

Framework of CBIR

HKBU Database Group

Framework of CBIR

- BOW Coding + Inverted File

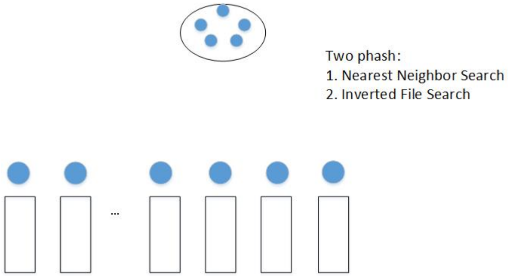


Figure 1: The CBIR framework we use

Framework of CBIR

- BOW Coding + Inverted File

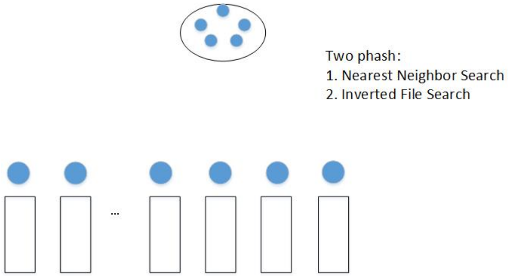


Figure 1: The CBIR framework we use

- BOW Coding: ANN (Approximate Nearest Neighbor)

Framework of CBIR

- BOW Coding + Inverted File

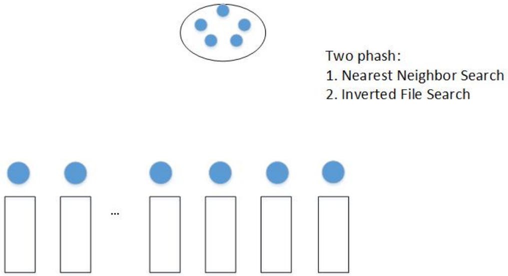


Figure 1: The CBIR framework we use

- BOW Coding: ANN (Approximate Nearest Neighbor)
  - Hierarchical k-means (HKM)

Framework of CBIR

- BOW Coding + Inverted File

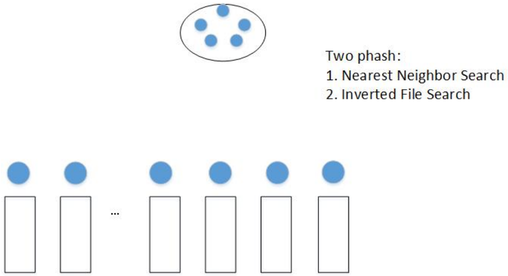


Figure 1: The CBIR framework we use

- BOW Coding: ANN (Approximate Nearest Neighbor)
  - Hierarchical k-means (HKM)
  - Approximate k-means (AKM)

Framework of CBIR

- BOW Coding + Inverted File

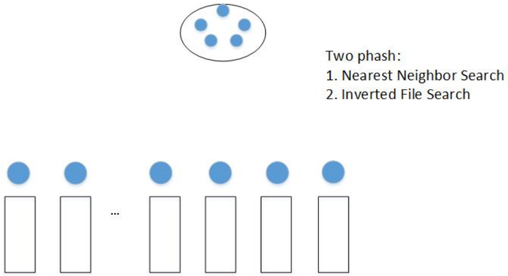


Figure 1: The CBIR framework we use

- BOW Coding: ANN (Approximate Nearest Neighbor)
  - Hierarchical k-means (HKM)
  - Approximate k-means (AKM)
- Authenticated CBIR

Framework of CBIR

- BOW Coding + Inverted File

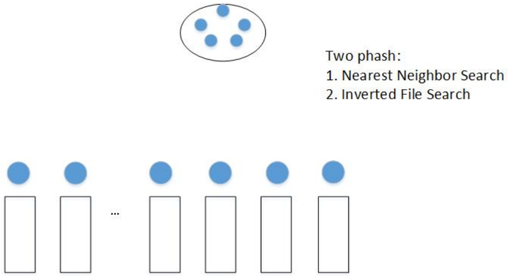


Figure 1: The CBIR framework we use

- BOW Coding: ANN (Approximate Nearest Neighbor)
  - Hierarchical k-means (HKM)
  - Approximate k-means (AKM)
- Authenticated CBIR
  - Should we outsource BOW coding? If yes, which algorithm should we choose? AKM (superior than HKM)

Framework of CBIR

- BOW Coding + Inverted File

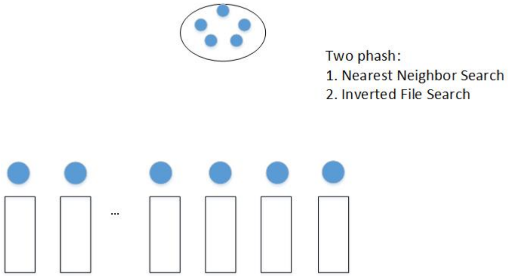


Figure 1: The CBIR framework we use

- BOW Coding: ANN (Approximate Nearest Neighbor)
  - Hierarchical k-means (HKM)
  - Approximate k-means (AKM)
- Authenticated CBIR
  - Should we outsource BOW coding? If yes, which algorithm should we choose? AKM (superior than HKM)
  - Authentication algorithms for both BOW coding(ANN) and similarity search

Problem Definition

- Entries: DO, SP and Client

Problem Definition

- Entries: DO, SP and Client
- Assumption: Client does not have enough storage or power for **BOW coding** and searching offline

Problem Definition

HKBU Database Group

- Entries: DO, SP and Client
- Assumption: Client does not have enough storage or power for **BOW coding** and searching offline
- Client sends SP a set of feature vectors extracted from a query image. SP then returns query results and VO, such that

Problem Definition

- Entries: DO, SP and Client
- Assumption: Client does not have enough storage or power for **BOW coding** and searching offline
- Client sends SP a set of feature vectors extracted from a query image. SP then returns query results and VO, such that
    - Client can verify the integrity of BOW coding

Problem Definition

- Entries: DO, SP and Client
- Assumption: Client does not have enough storage or power for **BOW coding** and searching offline
- Client sends SP a set of feature vectors extracted from a query image. SP then returns query results and VO, such that
  - Client can verify the integrity of BOW coding
  - (Make Sense) The VO size is much smaller than AKM data structure and computation complexity is reduced considerably compared with Bow coding offline

Problem Definition

- Entries: DO, SP and Client
- Assumption: Client does not have enough storage or power for **BOW coding** and searching offline
- Client sends SP a set of feature vectors extracted from a query image. SP then returns query results and VO, such that
  - Client can verify the integrity of BOW coding
  - (Make Sense) The VO size is much smaller than AKM data structure and computation complexity is reduced considerably compared with Bow coding offline
  - Client can verify the integrity of query results SP claims to return

Approximate k-means(AKM)

HKBU Database Group

- Random KD-trees: Build by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance

Approximate k-means(AKM)

- Random KD-trees: Build by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance

Approximate k-means(AKM)

- Random KD-trees: Build by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance

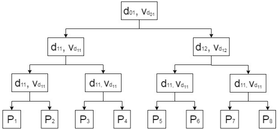


Figure 2: KD-tree

Approximate k-means(AKM)

HKBU Database Group

- Random KD-trees: Build by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance

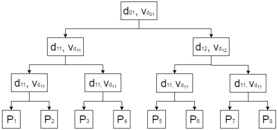


Figure 2: KD-tree

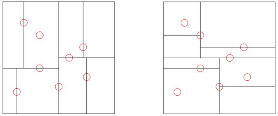


Figure 3: Random KD-trees example

Approximate k-means(AKM)

- Random KD-trees: Build by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance

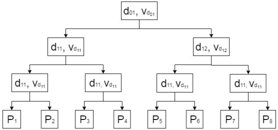


Figure 2: KD-tree

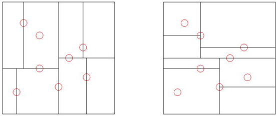


Figure 3: Random KD-trees example

- Approximate NN Search

Approximate k-means(AKM)

- Random KD-trees: Build by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance

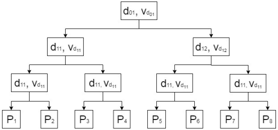


Figure 2: KD-tree

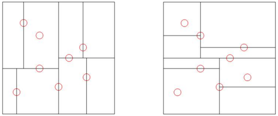


Figure 3: Random KD-trees example

- Approximate NN Search
  - When searching the trees, a single priority queue is maintained across all the randomized trees so that search can be ordered by increasing distance to each bin boundary.

Approximate k-means(AKM)

- Random KD-trees: Build by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance

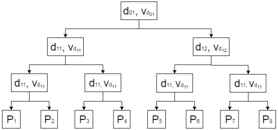


Figure 2: KD-tree

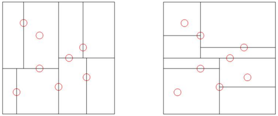


Figure 3: Random KD-trees example

- Approximate NN Search
  - When searching the trees, a single priority queue is maintained across all the randomized trees so that search can be ordered by increasing distance to each bin boundary.
  - The degree of approximation is determined by examining a fixed number of leaf nodes, at which point the search is terminated and the best candidates returned.

Authenticated AKM

HKBU Database Group

1. Framework of CBIR
2. Authenticated AKM
3. Authenticated Inverted File

Authenticated AKM

- Merkle random kd-trees: Just apply Merkle tree

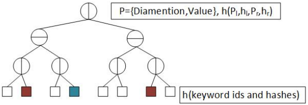
Authenticated AKM

- Merkle random kd-trees: Just apply Merkle tree
- Challenge: There are too many queries. It is inefficient if SP generates a VO for every query and then client verifies its integrity

Authenticated AKM

- Merkle random kd-trees: Just apply Merkle tree
- Challenge: There are too many queries. It is inefficient if SP generates a VO for every query and then client verifies its integrity
- Generate and verify VO in batches

VO Generation



VO: P[P[P[leafH leafN] P[leafH leafN]] P[(Pl nodeH) P[leafN leafH]]]

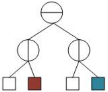
Figure 4: VO generation

Algorithm 1: Generate VO

Input: Location set L of leaf nodes, node P, VO;

if P is a leaf node **then**
    if P in L **then**
        VO $\leftarrow P$;
        return
    else
        VO $\leftarrow hash(P)$;
        return
    end if
end if
VO $\leftarrow P$;
VO $\leftarrow [$;
if P_l leafs $\cap L \neq \varnothing$ **then**
    GeneVO(L, P_l, VO);
else
    VO $\leftarrow P_l$;
end if
if P_r leafs $\cap L \neq \varnothing$ **then**
    GeneVO(L, P_r, VO);
else
    VO $\leftarrow P_r$;
end if
VO $\leftarrow]$;
return VO

Verification



VO: P[P[leafH leafN] P[leafH leafN]]

Figure 5: VO verification

Algorithm 2: Verify VO (computation of hash can be done synchronously)

Input: Queries Q_i, VO, lists H_i, current list locations, thresholds;

while VO still has entries **do**
  if P is an internal node **then**
    Split lists $H_{si} \leftarrow \varnothing$;
    for H_i **do**
      if The current threshold in H_i is smaller than the corresponding threshold **then**
        Split;
      else
        Location + 1;
      end if
    end for
    if The current thresholds in H_i are smaller than the corresponding thresholds **then**
      Current list locations + 1;
    else
      return Error;
    end if
  end if
  if [**then**
    VerifyVO(Q_{i_s}, VO, H_{i_s}, current list locations, thresholds);
  end if
  if] **then**
    return
  end if
end while
return

Verification

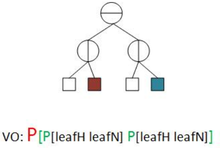


Figure 6: VO verification

Algorithm 3: Verify VO

Input: Queries Q_i, VO, lists H_i, current list locations, thresholds;

if P is leaf node **then**
  if The current thresholds in H_i are smaller than the corresponding thresholds **then**
    Current list locations + 1;
  else
    return Error;
  end if
end if
if P is leaf hash **then**
  if The current thresholds in H_i are greater than the corresponding thresholds **then**
    Current list locations + 1;
  else
    return Error;
  end if
end if

Check Leaf Nodes

- Challenge: False positive leaf nodes need to be checked

Check Leaf Nodes

- Challenge: False positive leaf nodes need to be checked
- Just reveal part of keyword for authentication

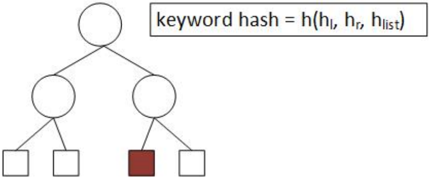


Figure 7: Keyword and its corresponding hash

Authenticated Inverted File

1. Framework of CBIR
2. Authenticated AKM
3. Authenticated Inverted File

Authenticated Inverted File

- Impact-Ordered VS. Frequency-Ordered

Authenticated Inverted File

- Impact-Ordered VS. Frequency-Ordered
  - Impact-Ordered: Less false positive but compression unfriendly

Authenticated Inverted File

- Impact-Ordered VS. Frequency-Ordered
  - Impact-Ordered: Less false positive but compression unfriendly
  - Frequency-Ordered: More false positive but compression friendly

Authenticated Inverted File

- Impact-Ordered VS. Frequency-Ordered
  - Impact-Ordered: Less false positive but compression unfriendly
  - Frequency-Ordered: More false positive but compression friendly

Authenticated Inverted File

- Impact-Ordered VS. Frequency-Ordered
  - Impact-Ordered: Less false positive but compression unfriendly
  - Frequency-Ordered: More false positive but compression friendly

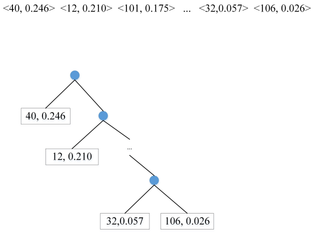


Figure 8: Impact sorted

Authenticated Inverted File

- Impact-Ordered VS. Frequency-Ordered
  - Impact-Ordered: Less false positive but compression unfriendly
  - Frequency-Ordered: More false positive but compression friendly

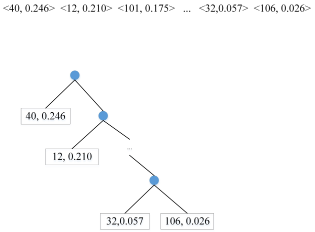


Figure 8: Impact sorted

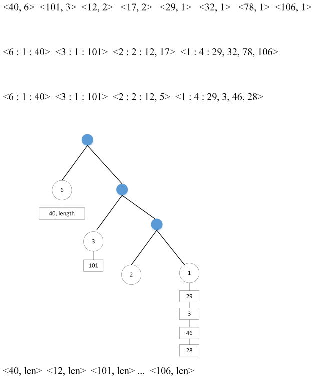


Figure 9: Frequency sorted

Authenticated Inverted File

- Combine Impact-Ordered and Frequency-Ordered

Authenticated Inverted File

- Combine Impact-Ordered and Frequency-Ordered

Authenticated Inverted File

- Combine Impact-Ordered and Frequency-Ordered

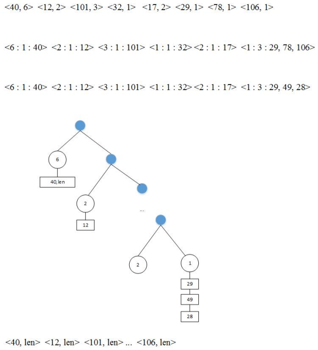


Figure 10: Combine impact-ordered and frequency-ordered 1

Authenticated Inverted File

- Combine Impact-Ordered and Frequency-Ordered

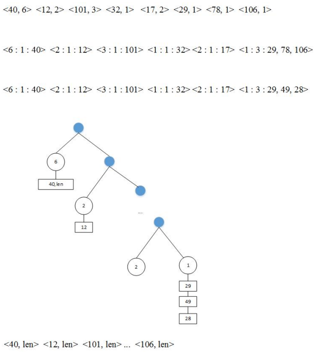


Figure 10: Combine impact-ordered and frequency-ordered 1

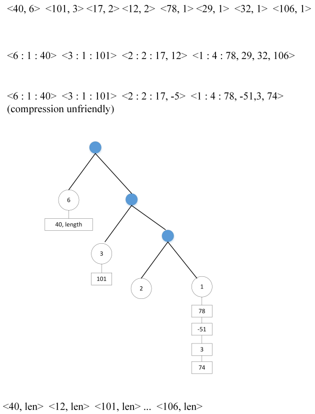


Figure 11: Combine impact-ordered and frequency-ordered 2

Other Issues

- Verification algorithm for HKM

Other Issues

- Verification algorithm for HKM
- Accuracy

Other Issues

- Verification algorithm for HKM
- Accuracy
    - Important in computer vision

Other Issues

- Verification algorithm for HKM
- Accuracy
  - Important in computer vision
  - Methods to increase accuracy in our framework

Other Issues

- Verification algorithm for HKM
- Accuracy
  - Important in computer vision
  - Methods to increase accuracy in our framework
    - Hamming embedding

Other Issues

- Verification algorithm for HKM
- Accuracy
  - Important in computer vision
  - Methods to increase accuracy in our framework
    - Hamming embedding
    - Geometric matching

Other Issues

HKBU Database Group

- Verification algorithm for HKM
- Accuracy
  - Important in computer vision
  - Methods to increase accuracy in our framework
    - Hamming embedding
    - Geometric matching
    - Query expansion

Other Issues

- Verification algorithm for HKM
- Accuracy
  - Important in computer vision
  - Methods to increase accuracy in our framework
    - Hamming embedding
    - Geometric matching
    - Query expansion
- Other frameworks of CBIR