Scalar Quantization for Large-Scale Image Search

Wengang Zhou[1], Yijuan Lu[2], Houqiang Li[3], and ***Qi Tian[1]***

[1]University of Texas at San Antonio, USA

[2]Texas State University at San Marcos, USA

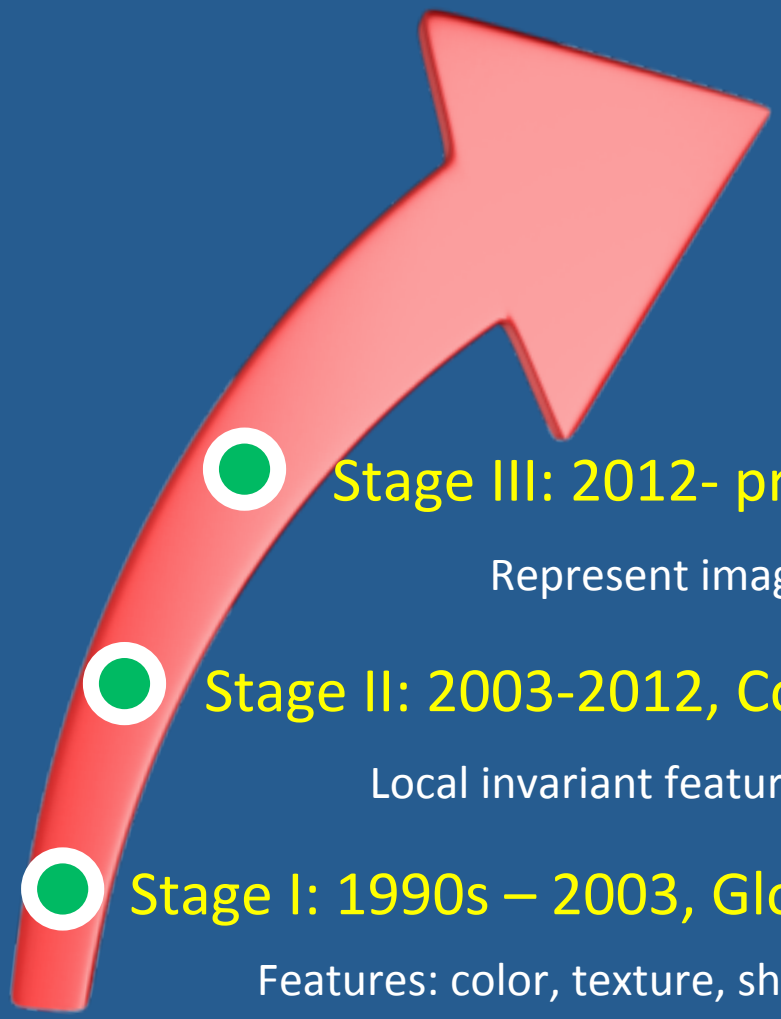
[3]University of Science and Technology of China




Outline

- Motivation
- Codebook-free CBIR
  - Scalar Quantization
  - Index Structure
  - Code Word Expansion
- Experiments
- Conclusion

Content based Image Retrieval



Stage III: 2012- present, Codebook-free CBIR

Represent images with binary features, no codebook training

Stage II: 2003-2012, Codebook -based CBIR

Local invariant features (SIFT), codebook for quantization

Stage I: 1990s – 2003, Global feature based CBIR

Features: color, texture, shape

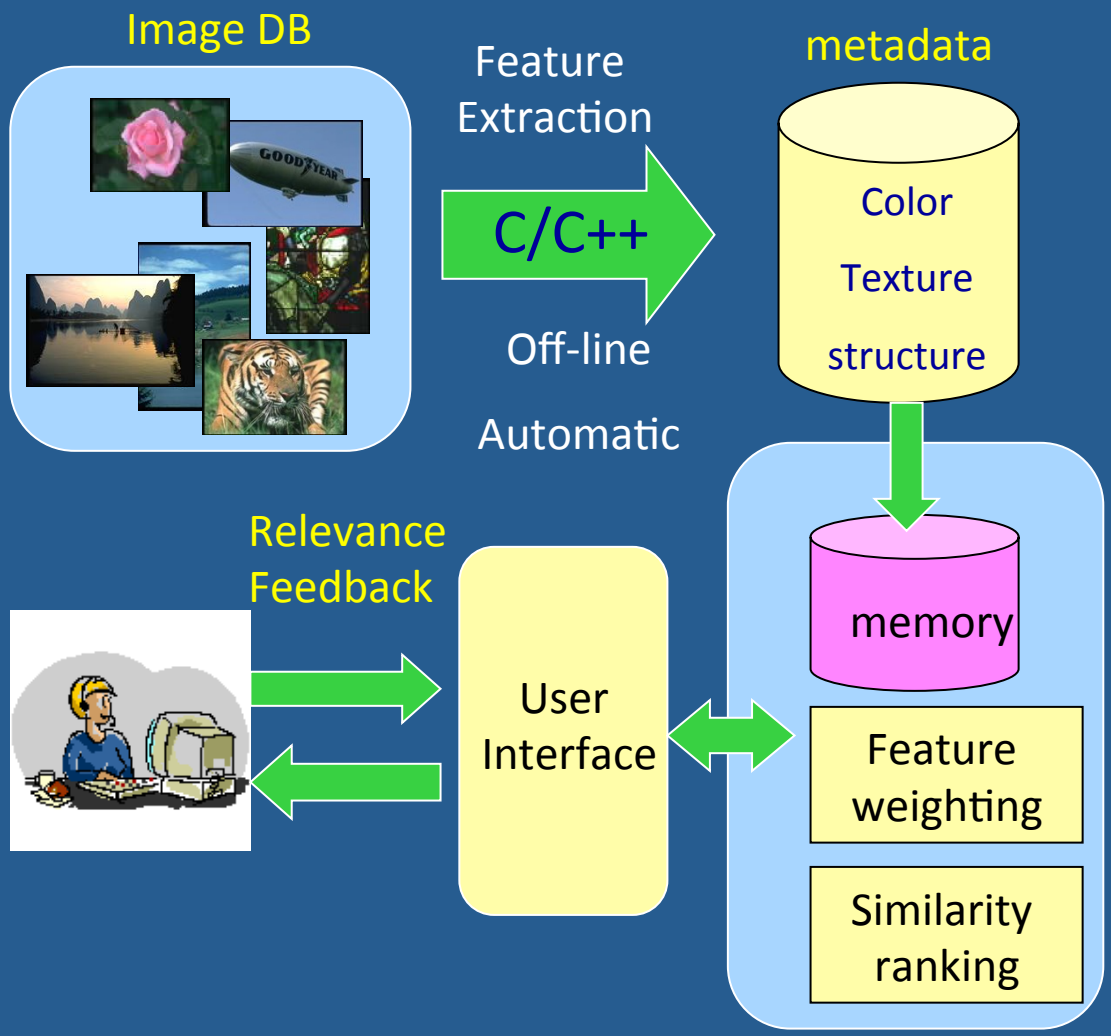
NN search

- Nearest neighbor search is inherently expensive due to the *curse of dimensionality*
- For high dimensions it turns out multi-dimensional indexing methods, such as the popular KD-tree are not more efficient than the brute- force exhaustive distance calculation
- There is a large body of literature on algorithms that overcome this issue by performing approximate nearest neighbor (ANN) search.
- For real data, LSH is outperformed by heuristic methods, which exploit the distribution of the vectors. These methods include randomized KD-trees and hierarchical k-means, both of which are implemented in the FLANN selection algorithm

NN search

- ANN algorithms are typically compared based on the trade-off between search quality and efficiency.
  - However, this trade-off does not take into account the memory requirements of the indexing structure.
- In the case of E2LSH (Euclidean Locality-Sensitive Hashing), the memory usage may even be higher than that of the original vectors.
- Both E2LSH and FLANN need to perform a final re-ranking step based on exact L2 distances, which requires the indexed vectors to be stored in main memory if access speed is important.
  - This constraint seriously limits the number of vectors that can be handled by these algorithms.
- Compact binary codes may reduce the problem

Stage I - Global Feature Based CBIR



- **Color**
  - Color histogram
  - Color moments
  - Color correlogram
- **Texture**
  - Tamura texture
  - Co-occurrence matrices
  - Gabor features
  - Wavelet moments
- **Shape**
  - Fourier descriptor
- Structure
  - Edge-based features

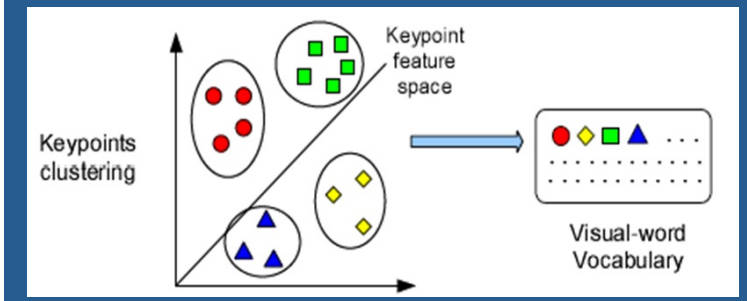
Rui & Huang, *et al.*, Relevance Feedback – a powerful tool for interactive content-based image retrieval, TCSVT, 1998

Large Scale Image Retrieval based on Bag-of-Visual-Word Model

Images

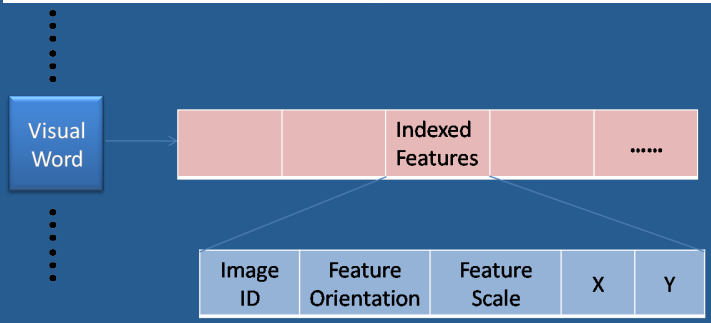
Local features

SIFT, SURF, MSER



Visual Codebook

Clustering by Vector Quantization:
- **K-means / hierarchical K-means**
- **Random forest**

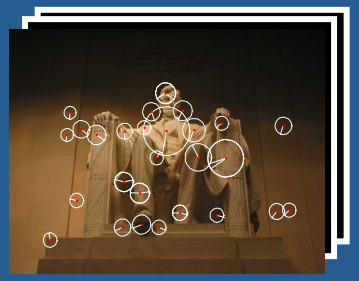


Index Structure

Inverted index file
Hashing

Image Retrieval

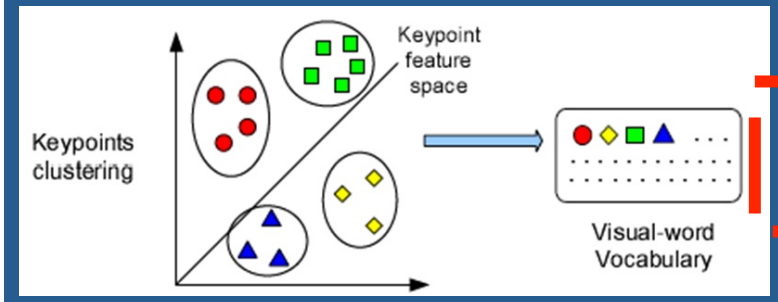
Codebook-based CBIR



Images

Local features

SIFT, SURF, HOG

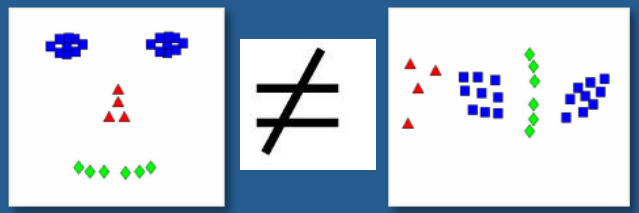


Visual Codebook

Clustering by Vector Quantization:
- **K-means / hierarchical K-means**
- **Random forest**

Index Structure

Inverted index file
Hashing

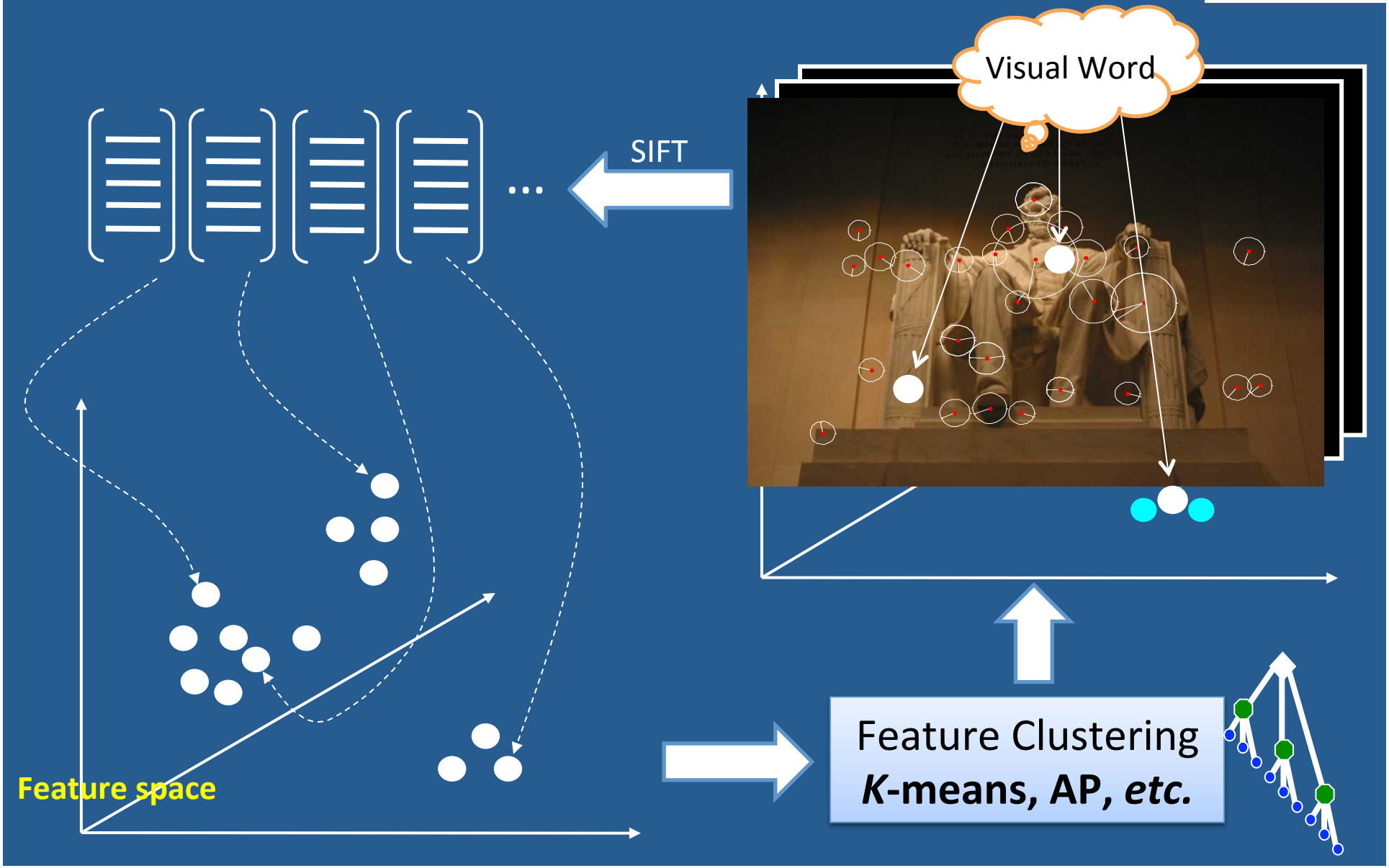


Spatial Verification

Local geometric verification
Global geometric verification

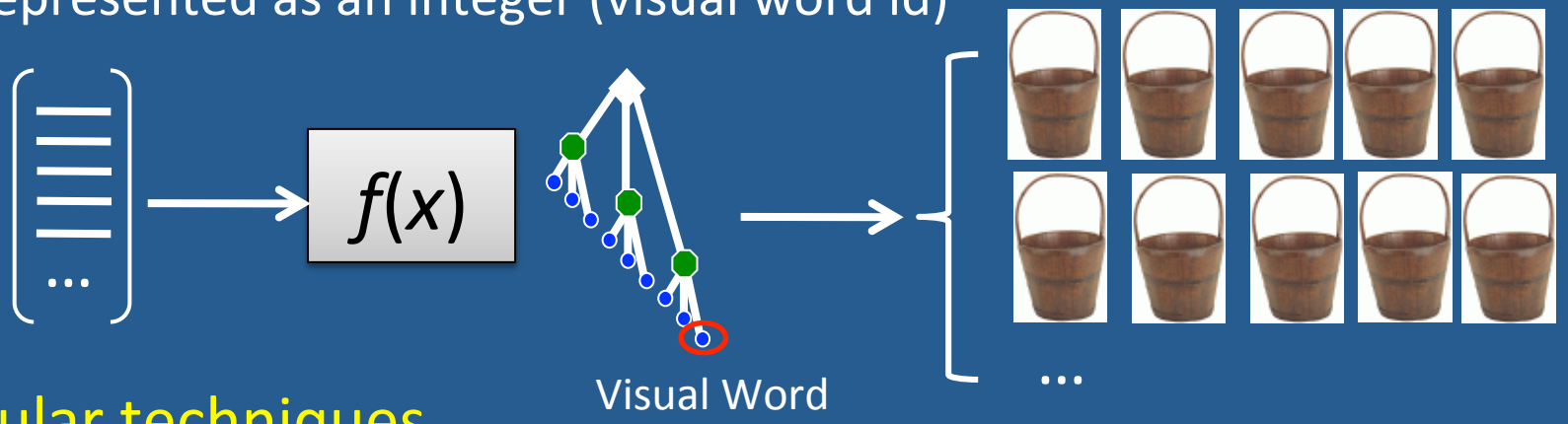
Application

Traditional Visual Codebook Generation

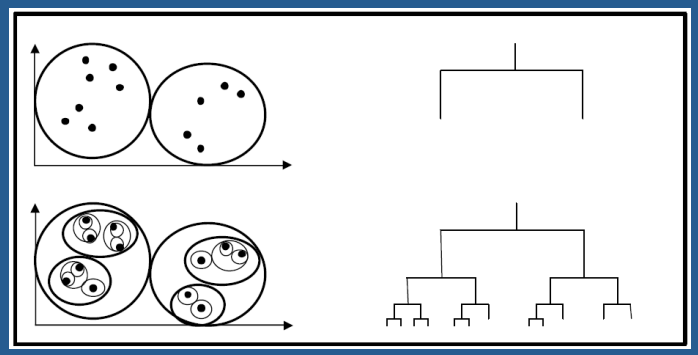


Vector Quantization

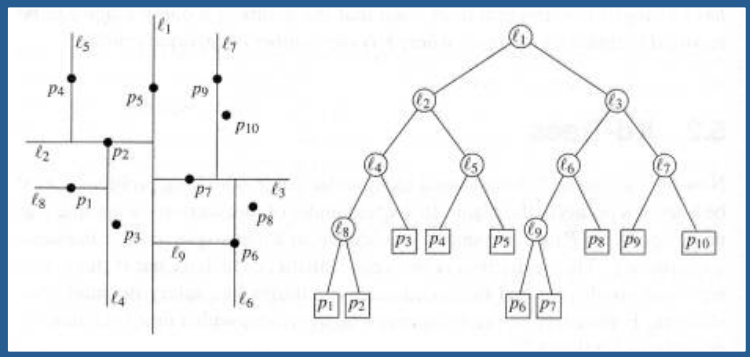
- From feature to visual word
  - Map a high-dimension vector to the closest visual word represented as an integer (visual word id)



- Popular techniques
  - Hierarchical 1-NN



K-D tree



Drawbacks of Vector Quantization

- High Computational Cost to train codebook
  - Codebook generation is computationally expensive, especially with a large amount of features, e.g. 1 billion
- Limited Reliability
  - Highly depend on training image database
  - Difficult to determine codebook size, e.g., 100K or 1M
  - Cannot well cover the feature space of indexed images
- Update Inefficiency
  - Difficult to update the codebook for newly collected images
  - Computational cost is high to update each visual word

Outline

- Motivation
- Codebook-free CBIR
  - Scalar Quantization
  - Index Structure
  - Code Word Expansion
- Experiments
- Demo
- Conclusion

Stage III: Codebook-free based CBIR

- Key Steps:

  Step 1: Convert a floating-point SIFT feature to a binary signature

  - Scalar Quantization vs. Vector Quantization

  Step 2: Adopt the classic inverted file indexing

  Step 3: Reduce quantization loss by code word expansion

Scalar Quantization

- Basic Idea
  - Scalar vs. Vector Quantization
    - simple, fast, data independent
  - Map a SIFT feature to a binary signature (bits)
    - Map function is independent of image collection
  - The binary signature keeps the discriminative power of SIFT feature

Step 1: Binary SIFT Signature

- General idea

Distance preserving

SIFT descriptor

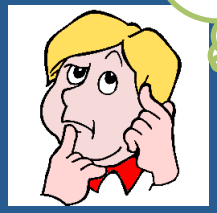
$(0, 25, 8, 2, \ldots, 14, 5, 2)^T$

Transformation $f(x)$

Binary Signature

$(0, 1, 0, 0, \ldots, 1, 0, 0)^T$

How to select $f(x)$



- Compact for storing in memory
- Efficient for comparison computing

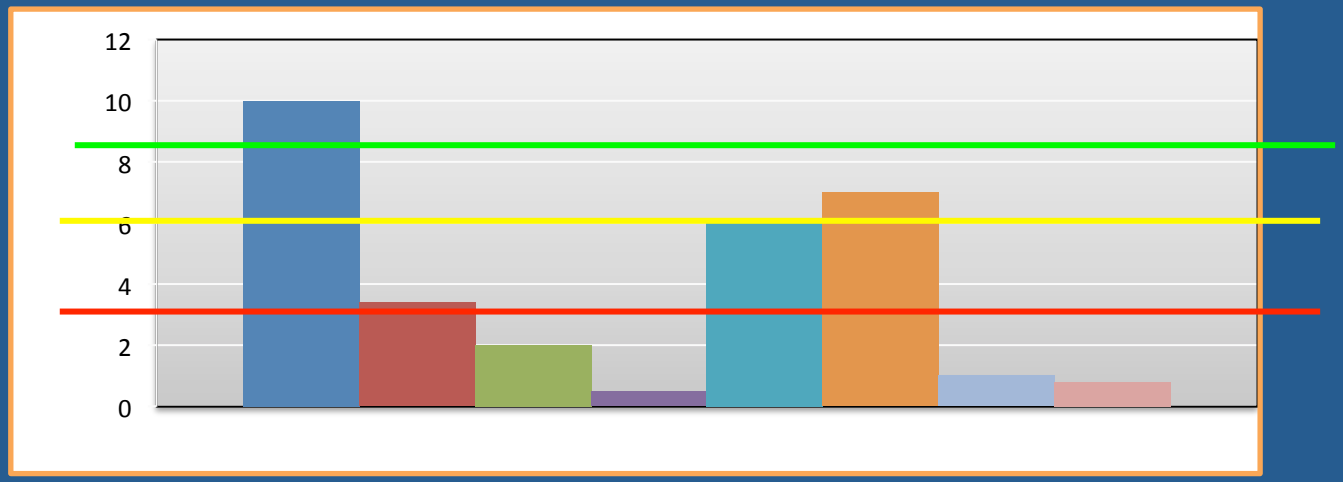
Preferred Properties

- Simple and effective
- Unsupervised to avoid overfitting to training data
- Well preserved feature distance

Binary SIFT Signature

- Given a SIFT descriptor $f = (f_1, f_2, \cdots, f_d)^T \in R^d$
- Transform it to a bit vector $b = (b_1, b_2, \cdots, b_{k \cdot d})^T$
  - Each dimension is encoded with k bits, $k \leqslant \log_2 d$

Example:
d = 8



1	1	0	0	**1**	1	0	0
1	**0**	**0**	**0**	**0**	**1**	**0**	**0**
1	**0**	**0**	**0**	**0**	0	**0**	**0**

Scalar Quantization

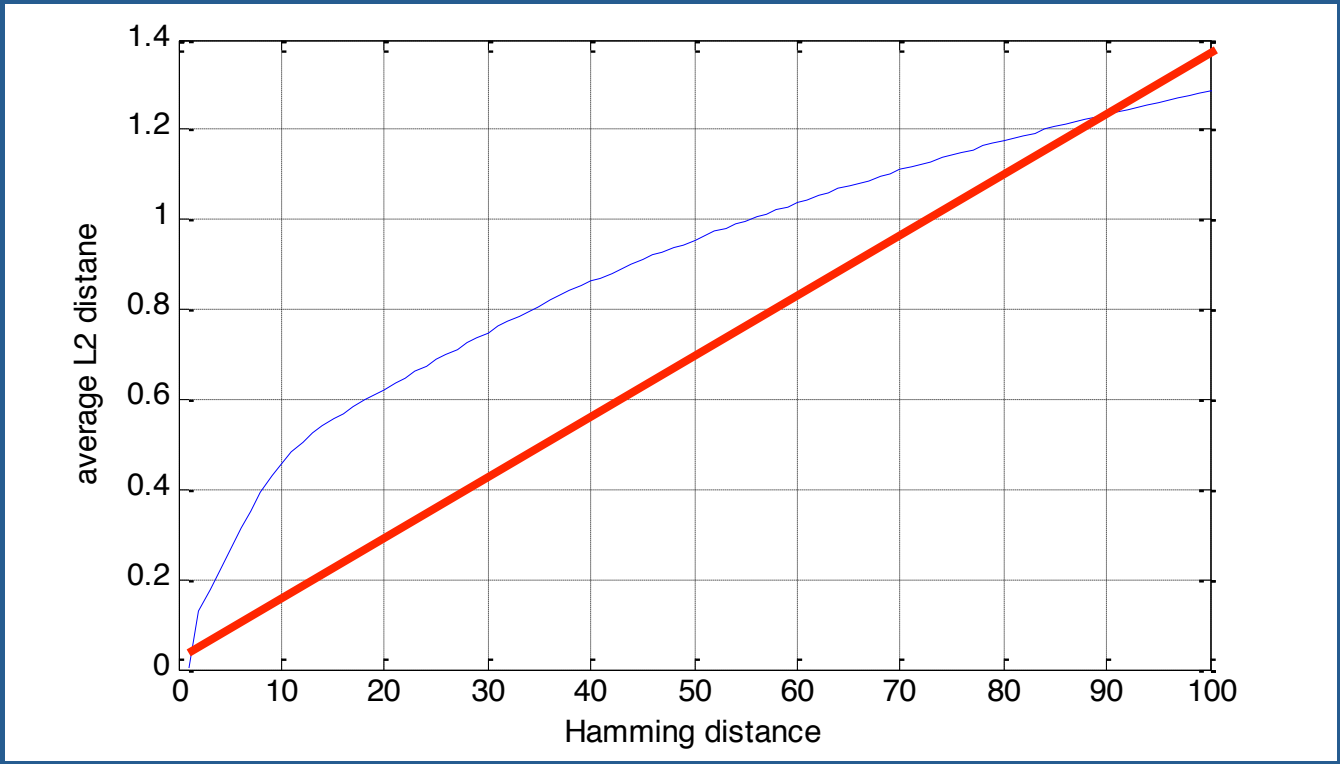
- To start, each dimension is encoded with one bit
  - Given a feature vector $f = (f_1, f_2, \cdots, f_d)^T \in R^d$
  - Quantize it to a bit vector $b = (b_1, b_2, \cdots, b_d)^T$

$$b_i = \begin{cases} 1 & \text{if } f_i > \hat{f} \\ 0 & \text{if } f_i \le \hat{f} \end{cases} \quad (i = 1, 2, \cdots, d)$$

$\hat{f}$ The median value of vector f

Experimental Observation

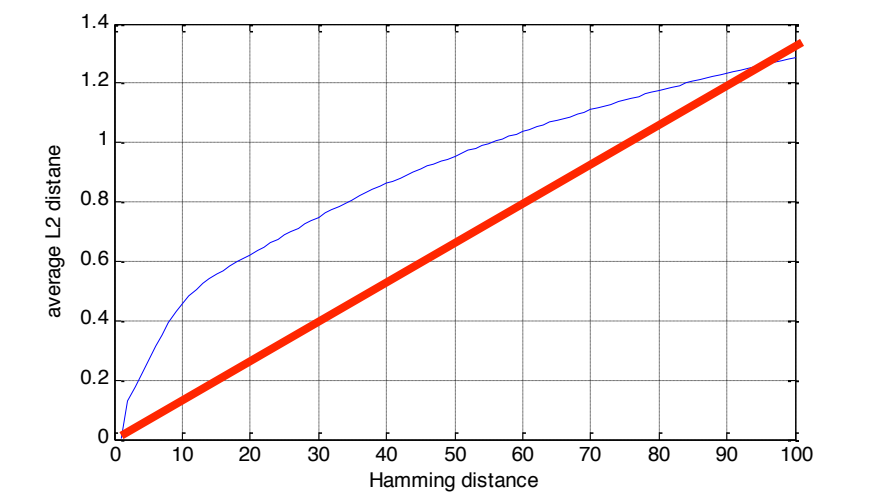
Statistical study on 0.4 trillion feature pairs



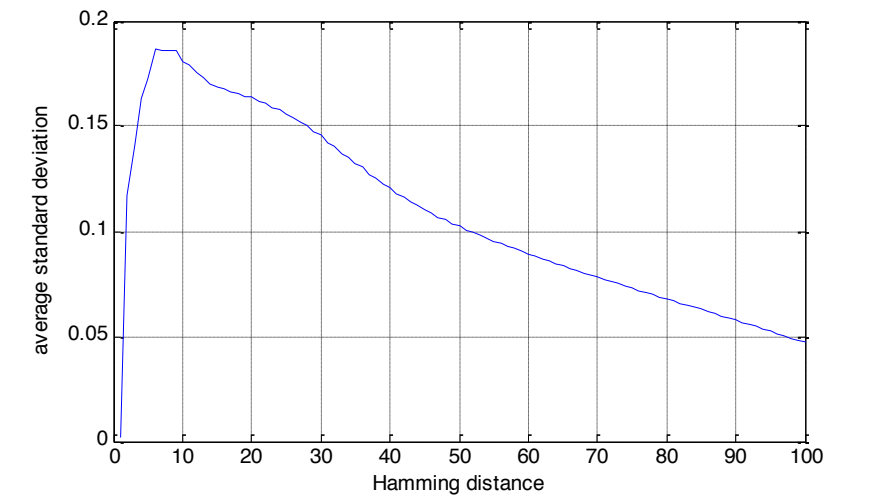
Euclidean distance vs. Hamming distance;

Experimental Observation

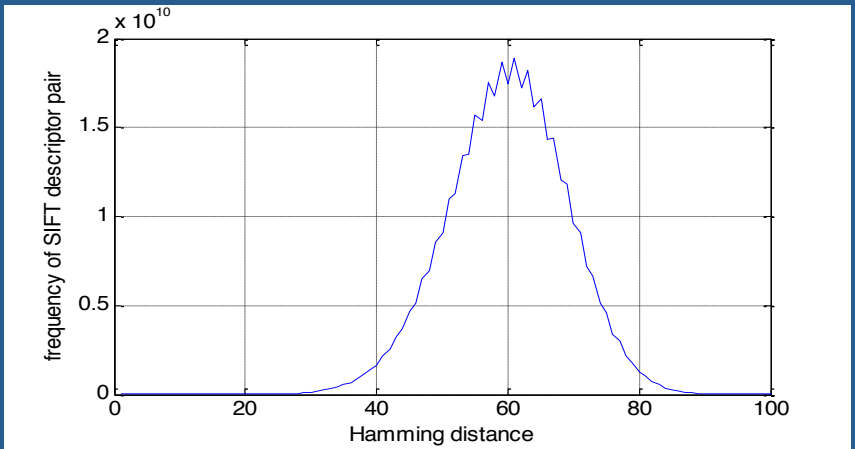
Statistical study on 0.4 trillion feature pairs



(a) **Euclidean distance vs. Hamming distance;**

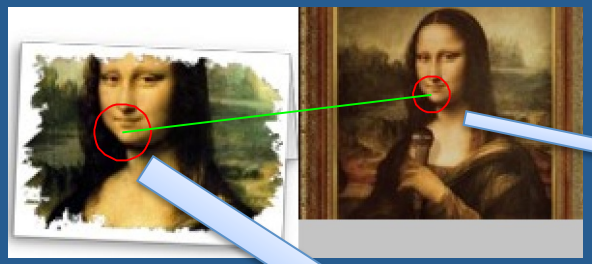


(b) **The average standard deviation vs. Hamming distance.**



(c) **Descriptor pair frequency vs. Hamming distance;**

An Example of Matched Features

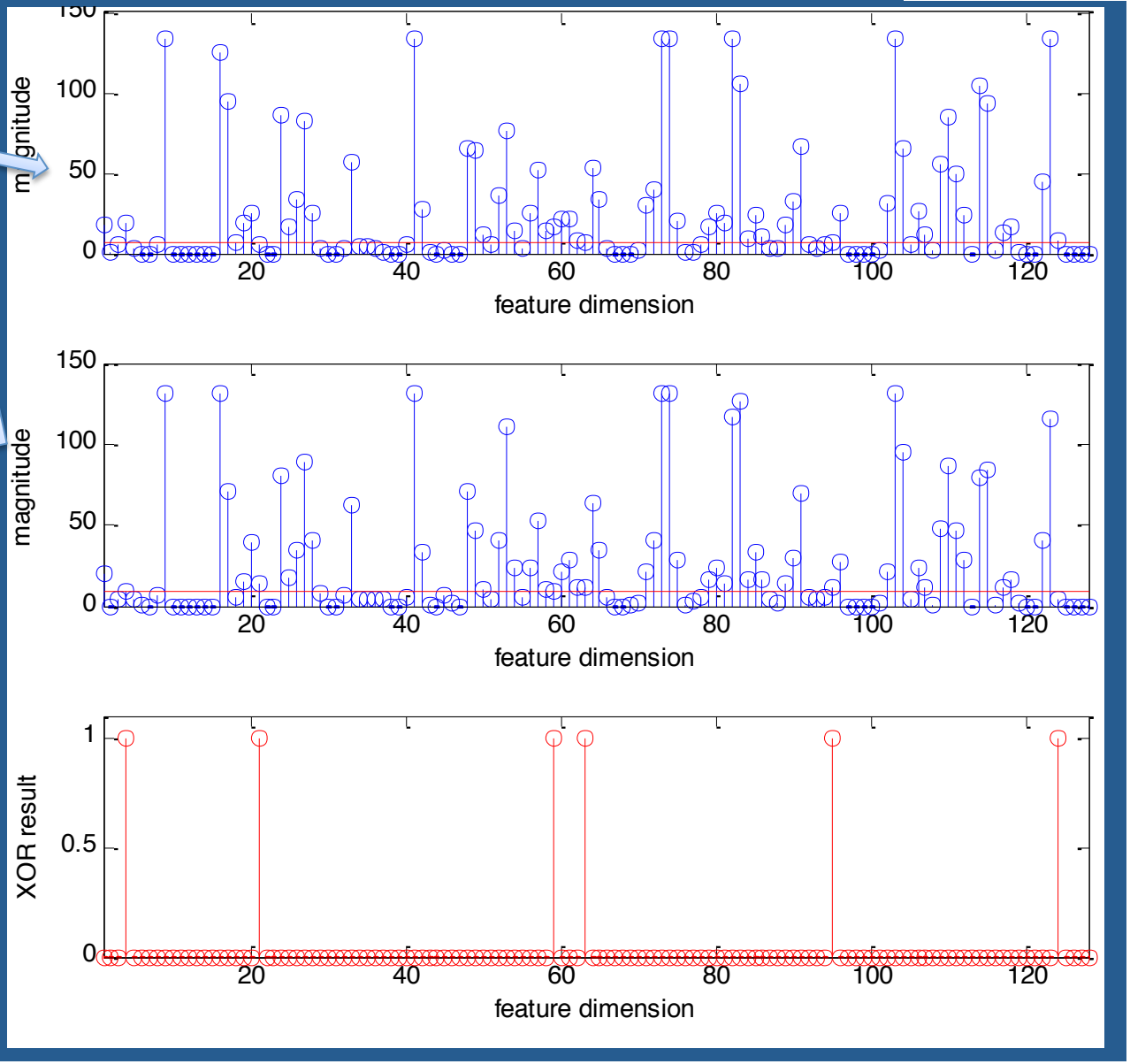


Observation

Share some common patterns in magnitudes on the 128 bins, e.g., the pair-wise differences between most of bins are similar and stable.

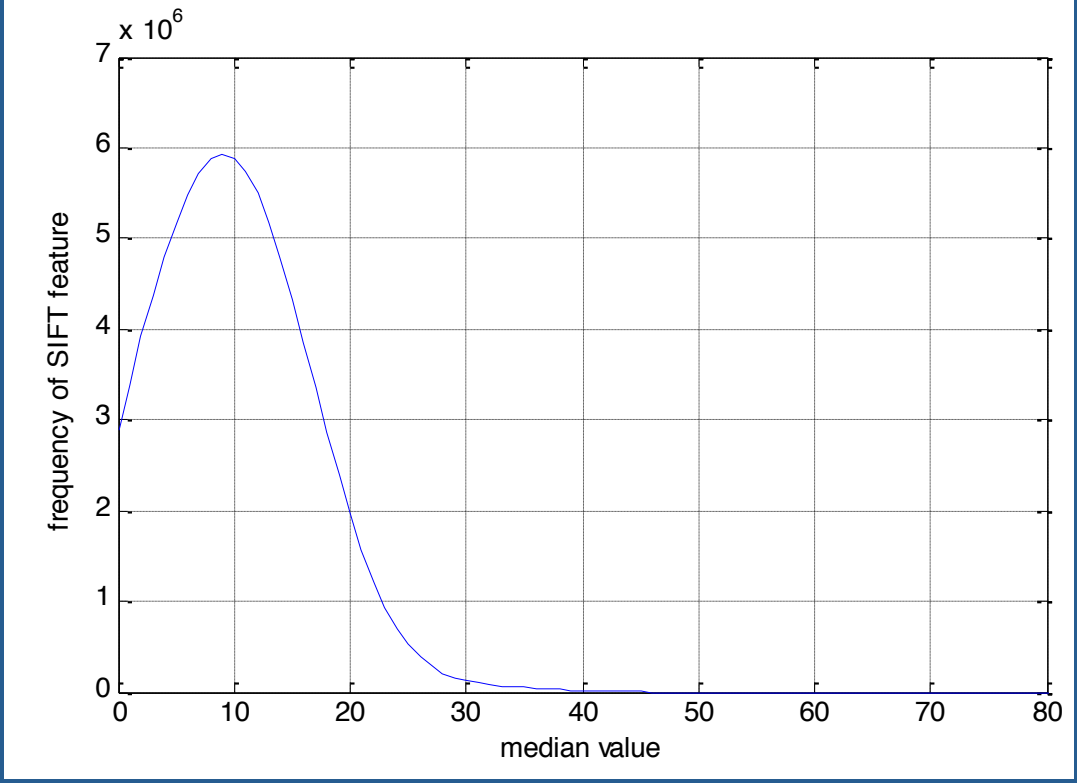
Implication:

The differences between bin magnitudes and a predefined threshold are stable for most bins.



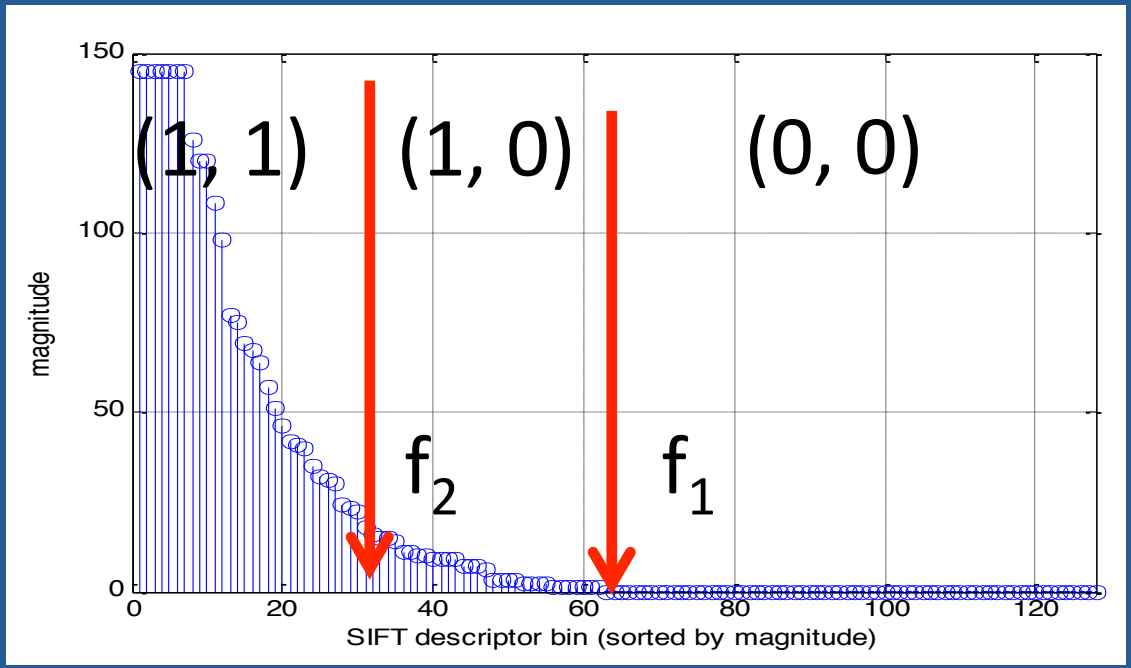
Distribution of SIFT Median Value

- Distribution on 100 million SIFT features



Scalar Quantization

- Generalize Scalar Quantization
  - Encode each dimension with multi-bits, e.g., 2 bits
  - Trade-off between memory cost and accuracy



A typical SIFT descriptor with bin magnitude sorted in descending order

Scalar Quantization

- Each dimension is encoded with two bits
  - In practice, we quantize each dimension with 2 bits
    - Considering memory and accuracy

$$(b_i, b_{i+128}) = \begin{cases} (1,\ 1) & \text{if } f_i > \hat{f}_2 \\ (1,\ 0) & \text{if } \hat{f}_1 < f_i \le \hat{f}_2 \\ (0,\ 0) & \text{if } f_i \le \hat{f}_1 \end{cases} \qquad (i = 1,2,\cdots,d)$$

where $\hat{f}_1 = \dfrac{g_{64} + g_{65}}{2}, \hat{f}_2 = \dfrac{g_{32} + g_{33}}{2}$

$(g_1, g_2, \cdots, g_{128})$ is descendingly sorted from $(f_1, f_2, \cdots, f_{128})$

Visual Matching by Binary Signature

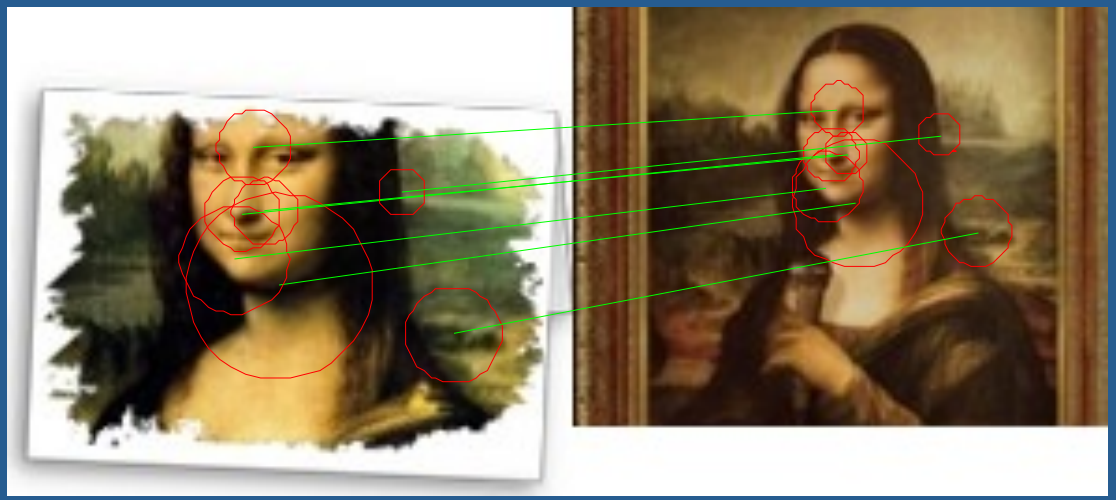
- Given SIFT $f^{(1)}$ from Image I_q and $f^{(2)}$ from image I_d
- Perform scalar quantization:

$$f^{(1)} \rightarrow b^{(1)}; \qquad f^{(2)} \rightarrow b^{(2)}$$

- $f^{(1)}$ matches $f^{(2)}$, if Hamming distance

$$d(b^{(1)}, b^{(2)}) < \textit{Threshold}$$

Real example:
256-bit SIFT binary signature
Threshold = 24 bits

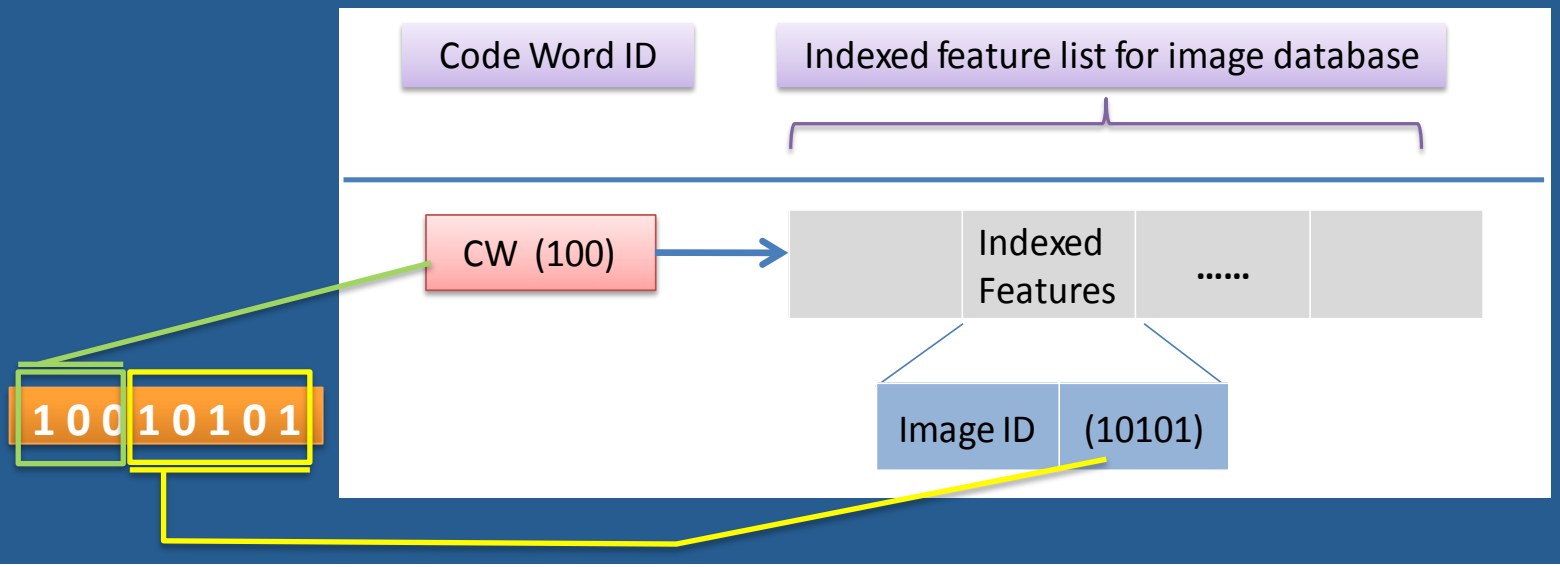


Outline

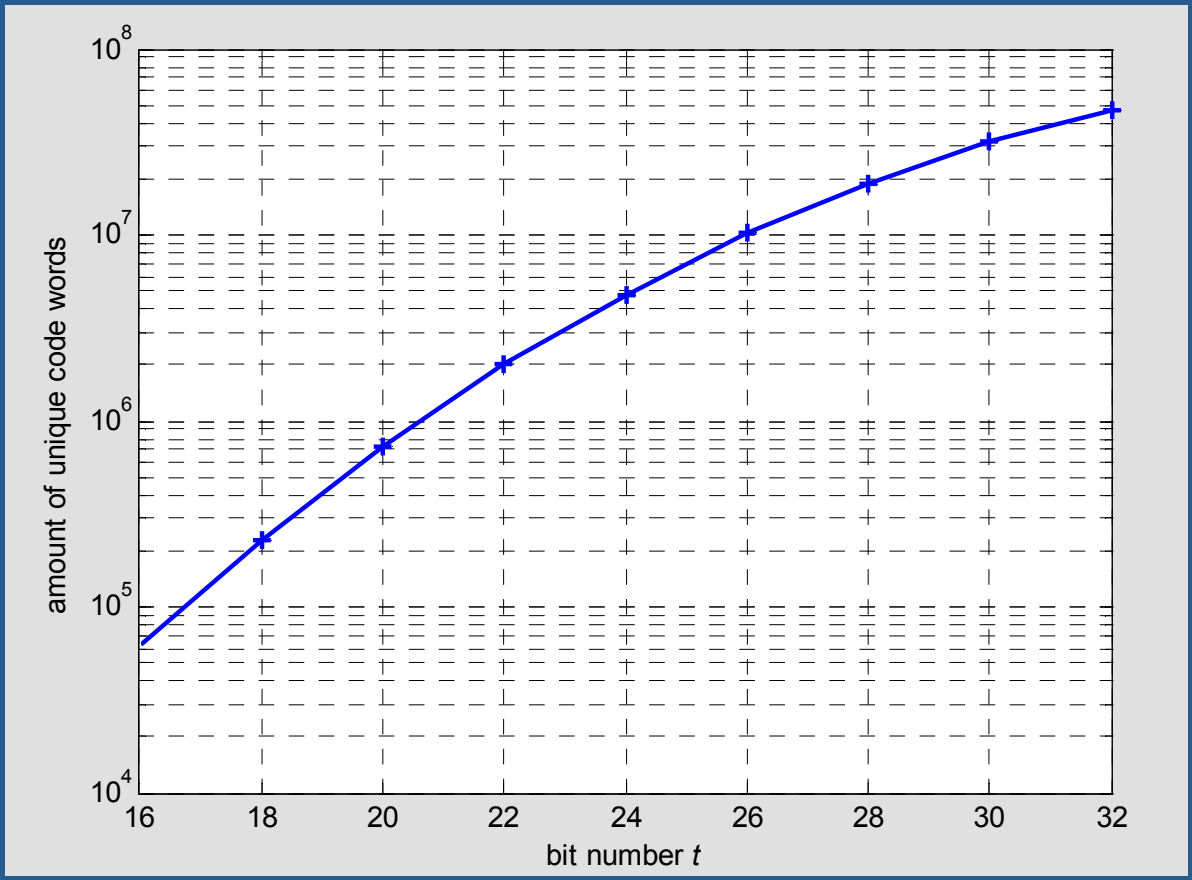
- Motivation
- Codebook-free CBIR
  - Scalar Quantization
  - Index structure
  - Code Word Expansion
- Experiments
- Demo
- Conclusion

Step 2: Indexing with Scalar Quantization

- Adopt the classic inverted file indexing
  - Fast, scalable
- Take the top *t* bits of the binary signature as "code word"
  - Codeword address
- Store the remaining bits in the entry of indexed features
- A toy example :



Indexing with Scalar Quantization

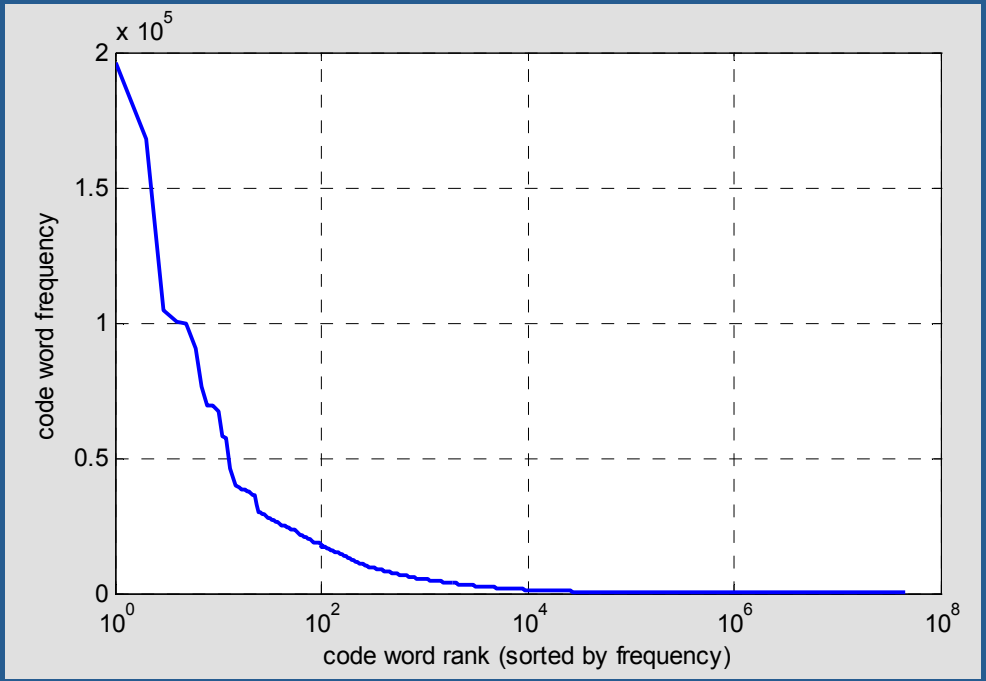


- **The amount of unique code words (top *t* bits from 256-bit vector) for different *t* on 1-million image database.**

Indexing with Scalar Quantization

- Generally, the more code words are generated, the shorter the average length of indexed feature list becomes, and the less the time cost is needed to query a new feature.
- What happens if there's a bit flip in the "code word" ? There's need of a sort of query expansion to search within more code words for each query feature. And the number of expanded indexed feature lists is polynomial to t .
- To make a tradeoff, we select $t = 32$, obtaining, in the experiments, 46.5 million "code words".

Indexing with Scalar Quantization



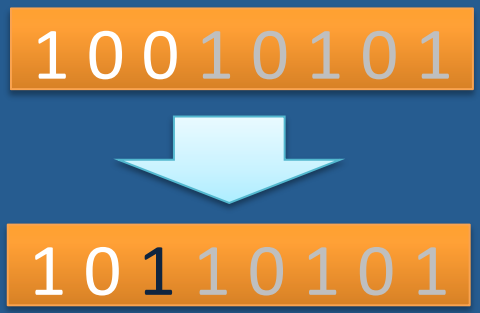
- The figure shows the distribution of code word occurrence on one million image database.
- Only the top few thousand code words have very high frequency. These code words are prevalent in many images, and their distinctive power is weak.
- A stop-list is used to ignore those high frequency code words that occur in more than 0.11% of the total image dataset.
- Experiments reveal that a proper stop-list may not affect the search accuracy, but does avoid checking many code word lists and achieves gain in efficiency.

Outline

- Motivation
- Our Approach
  - Scalar Quantization
  - Index Structure
  - Code Word Expansion
- Experiments
- Demo
- Conclusion

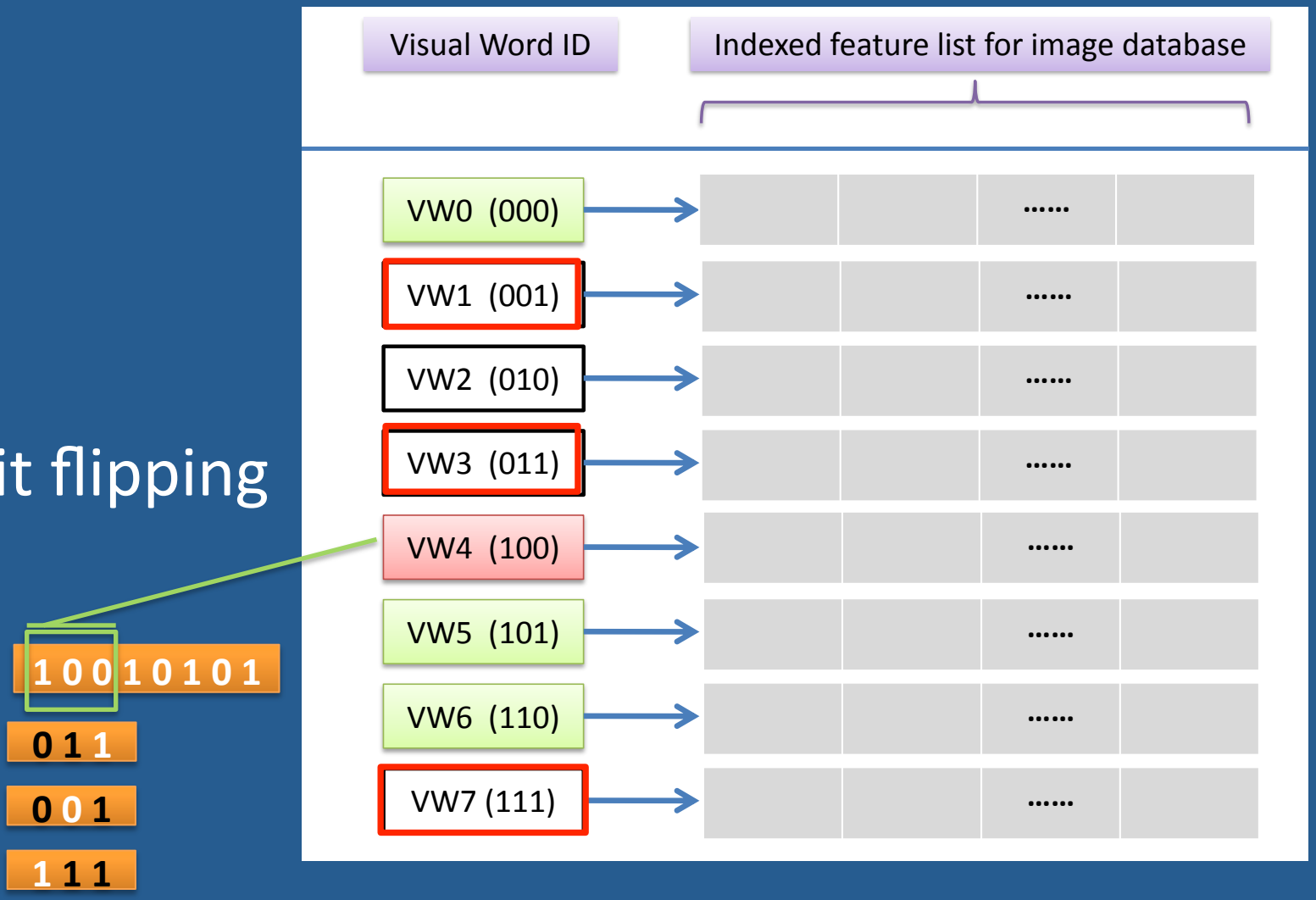
Step 3: Code Word Expansion

- Quantization Error
  - Flipping bits exist in code word



  - If ignore those flipped bits, many candidate features will be missed
    - **Degraded recall !!**
  - Solution: Expand code word to include flipped code words
    - Enumerate all possible nearest neighbors within a predefined Hamming distance

Step3: Code Word Expansion
- Quantization Error Reduction



Outline

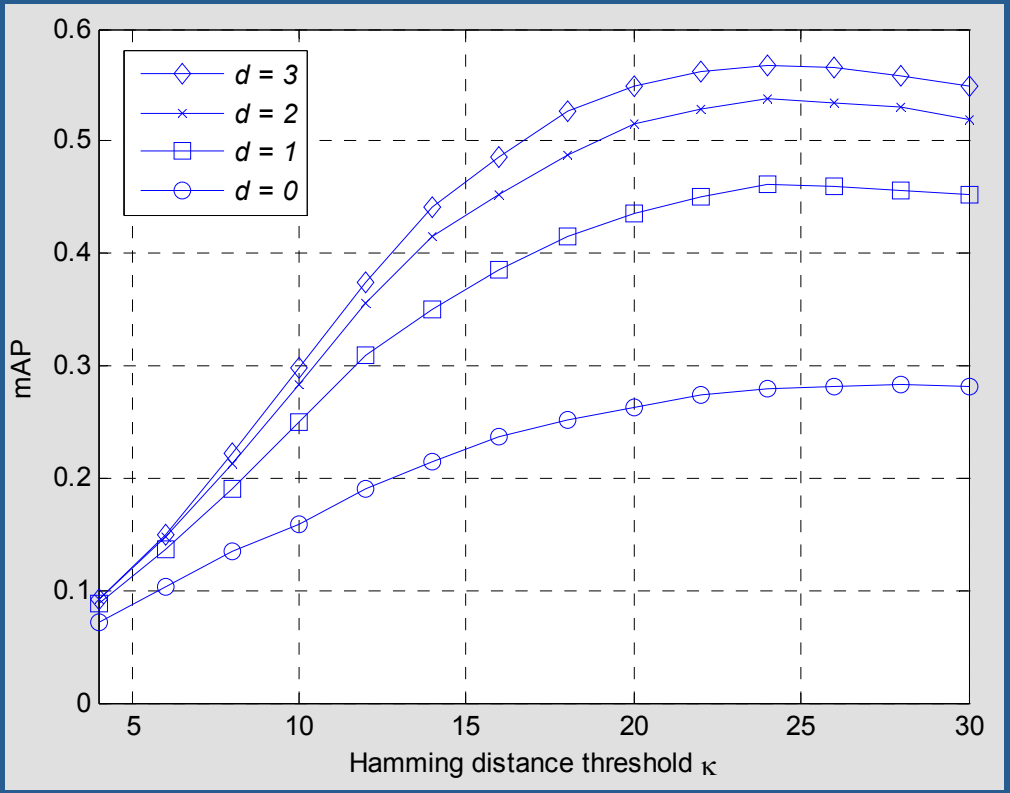
- Motivation
- Our Approach
  - Scalar quantization
  - Index structure
  - Code word expansion
- Experiments
- Conclusion
- Demo

Experimental Results

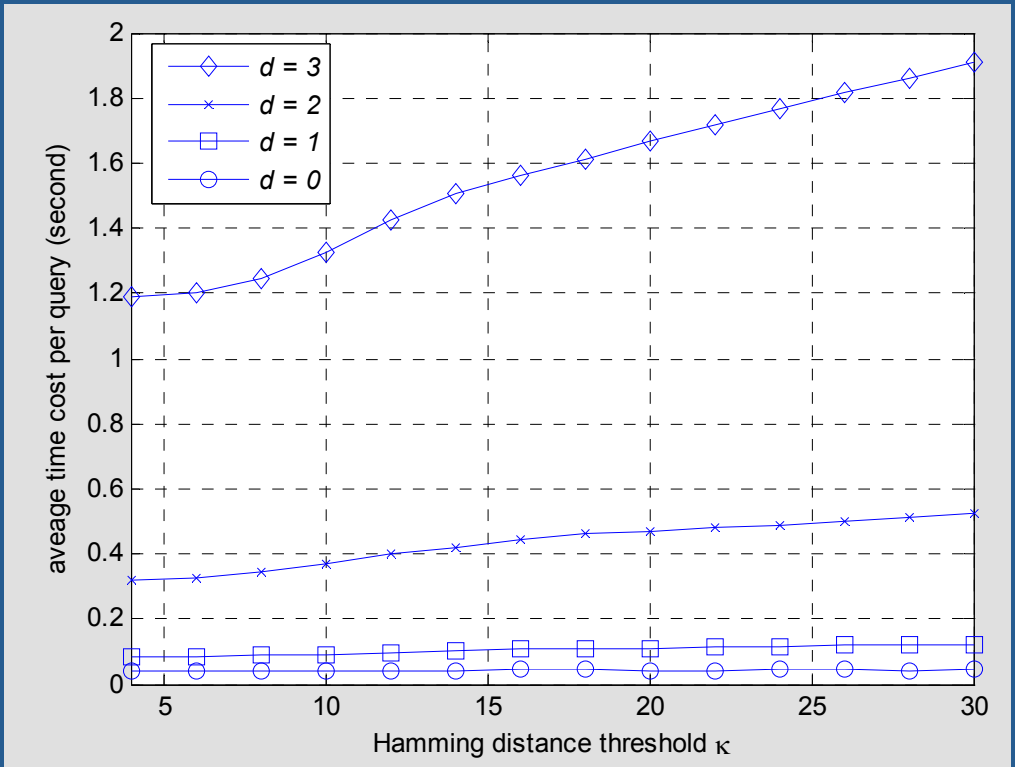
- Experiment Setup
  - Basic dataset:
    - One million images crawled from the Web
  - Ground truth dataset
    - Partial-duplicate dataset of 1103 images
    - 33 groups, containing logo, artwork, trademark, etc.
    - Available for download (if the website of Wengang Zhou comes back online…)
  - Comparison approaches
    - Baseline (visual vocabulary tree), Nister, CVPR'06
    - Hamming Embedding, Jegou, ECCV'08
    - Soft Assignment, Philbin, CVPR'08
  - Computer configuration
    - CPU 3.4G Hz, 16G memory

Parameter Analysis

- There are two parameters in our approach: Hamming distance threshold κ and expansion-bit number *d*.
- To study the impact of these two parameters on search performance and computational cost, we compare the mAP performance and average time cost per query under different parameter settings of κ and *d* on the 1-M image dataset.

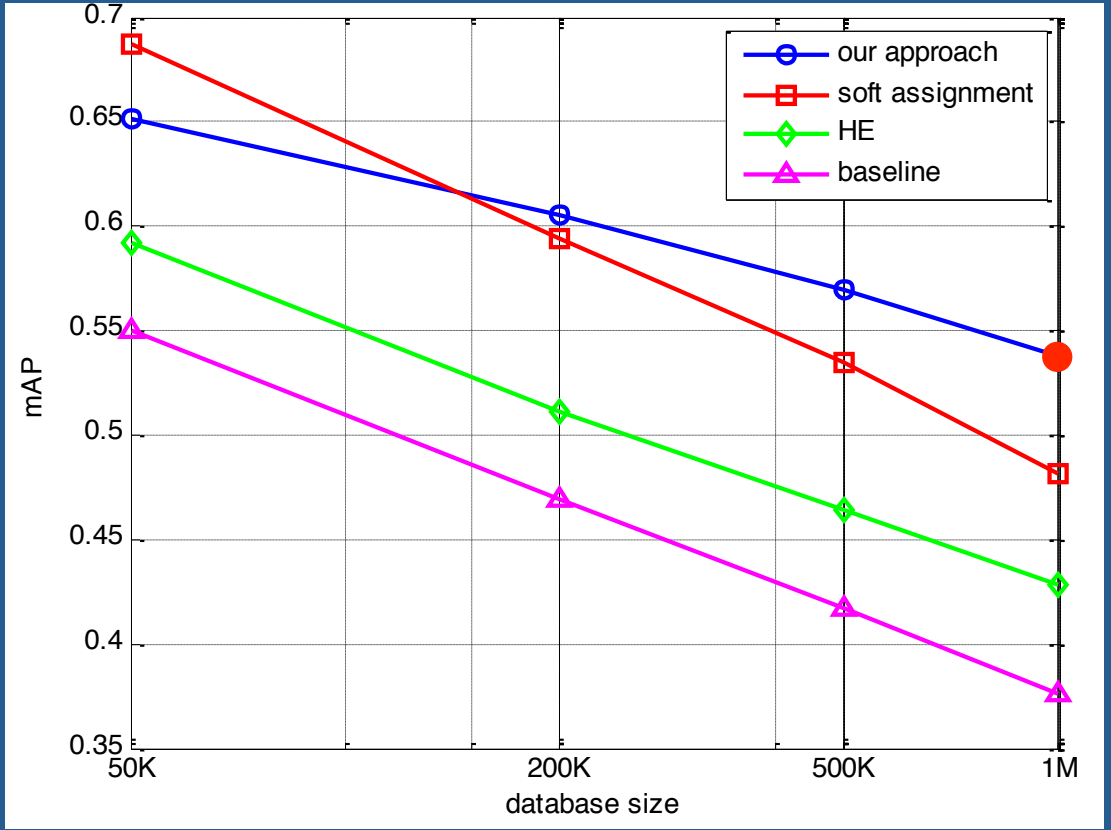


- When the Hamming distance threshold κ increases, the mAP performance first increases and then keeps stable and gradually drops a little after it reaches the peak, where $\kappa = 24$.
  - This is intuitive, since increasing κ always includes more candidate true matches, but when κ is too large, many noisy matches are also included.
- When expansion-bit number *d* increases, the mAP gradually increases. This is due to the fact that more candidate code word lists are involved in matching verification, and more true matches will be kept.



- The average time cost per query increases when κ increases. This is due to that, when κ is larger, we have to make more exclusive-OR operations, until the Hamming distance between two 224-bit vectors is above a threshold.
- As *d* increases, the querying time cost rises significantly. This is because the expanded code word list number is exponential to the expansion-bit number *d*.

Comparison: Accuracy



mean Average Precision (mAP) on 1-M image dataset

mAP improvement: (our approach: 0.54)

Baseline: 0.38, relatively 42.1% improvement

HE: 0.43, 25.6% improvement

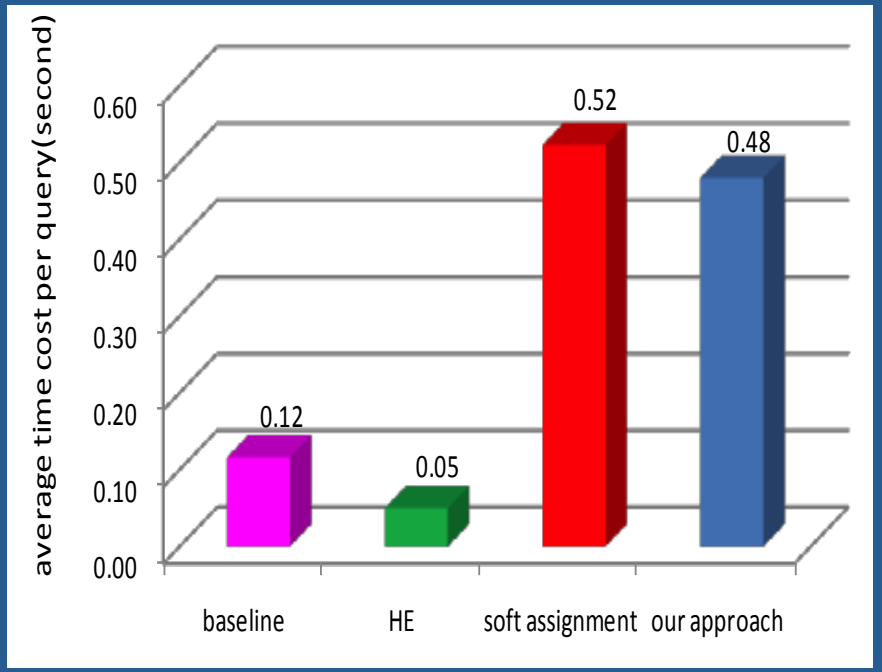
Soft: 0.48, 12.5% improvement

Comparison: Efficiency

Off-line indexing
(Time cost to index 1-million SIFT)

Method	Time cost (seconds)
baseline	53.72
HE	64.82
soft assignment	771.09
our approach	**18.86**

On-line retrieval (Average time cost per query on 1-M image dataset)



0.48 seconds per query on 1-M dataset

Comparison: Memory Cost

	Memory cost per indexed feature (Bytes)	Memory cost for online quantization (Bytes)
baseline	8	142M
HE	12	398M
soft assignment	24	506M
our approach	**32**	**0**

Sample Results

Query

Outline

- Motivation
- Codebook-free CBIR
  - Scalar Quantization
  - Index Structure
  - Code Word Expansion
- Experiments
- Demo
- Conclusion

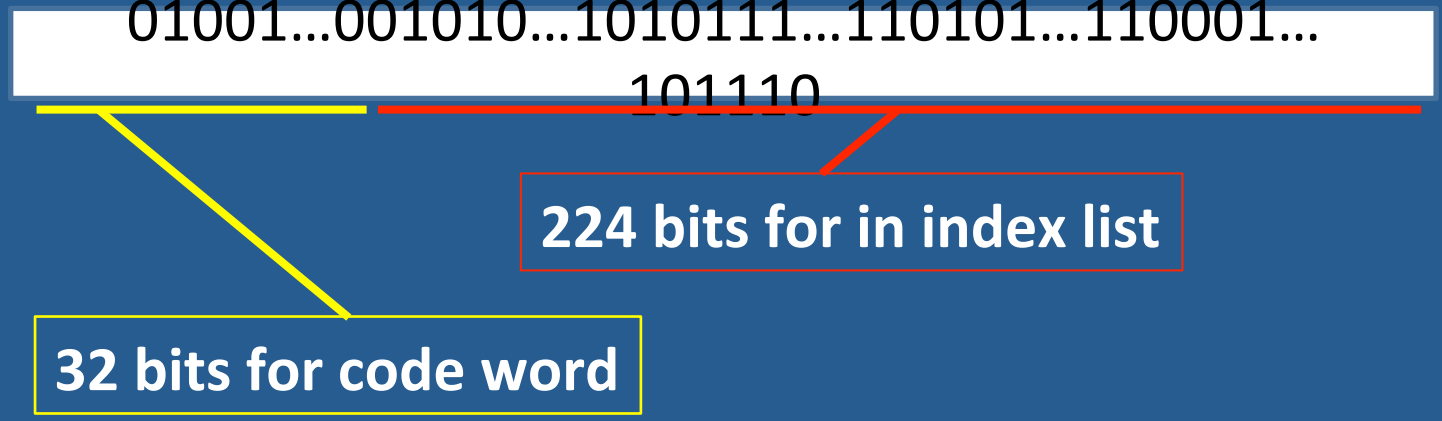
Conclusion & Discussion

- Contributions
  - A novel **codebook-free CBIR framework**
    - Scalar Quantization to convert SIFT descriptor to binary signature
      - Image-collection independent;
      - Efficient to implement
    - Adapt binary signature to inverted index structure for large-scale image search
    - Reduce quantization loss by code word expansion
- Future Work
  - Threshold selection in scalar quantization
    - Learning-based vs. median
  - Dimension reduction of binary signature
    - More compact
  - Apply Scalar Quantization to general features and mobile apps

Thanks! ☺

Any Questions?

Analysis on Recall of Valid Features



Retrieved features as candidates:

$$S = \{HamDis_{32} \leq 2\}$$

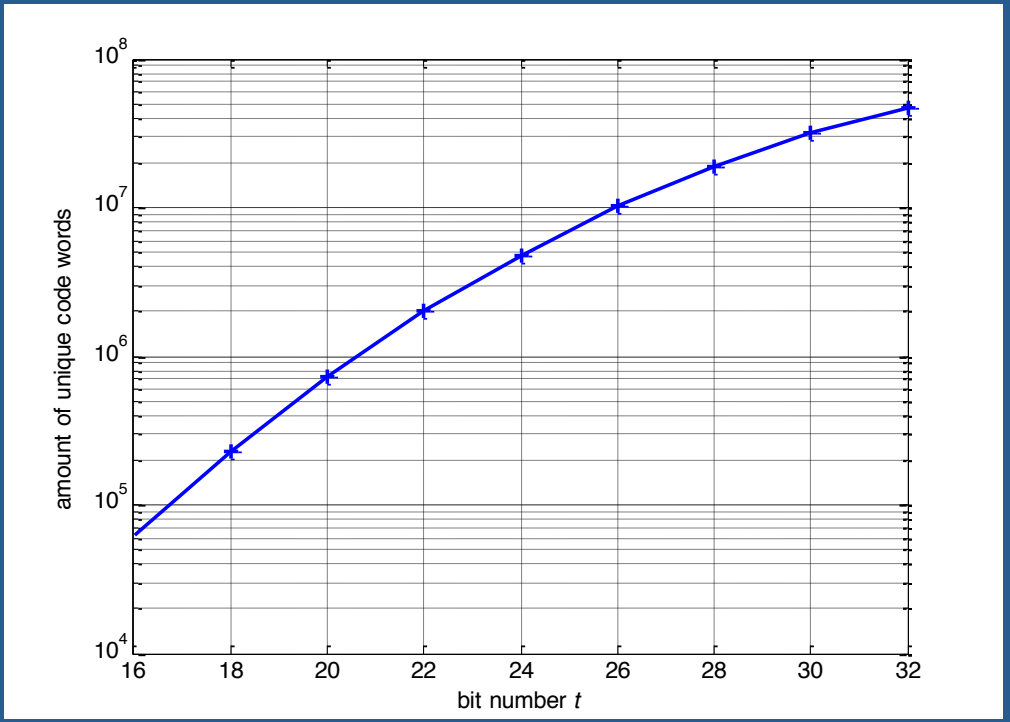
All candidate features :

$$\Omega = \{HamDis_{256} \leq 24\}$$

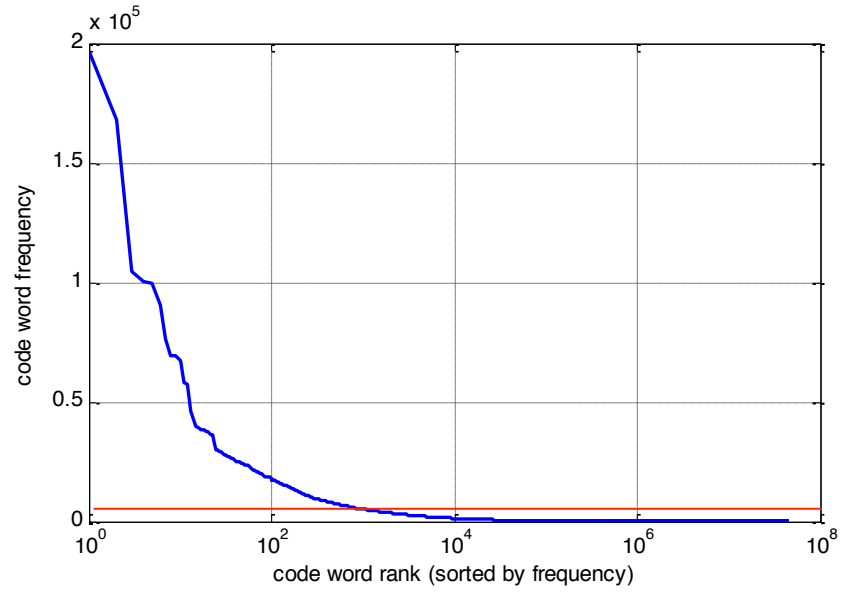
$$recall = \frac{S \cap \Omega}{\Omega} = 39.8\%$$

Unique “Code Word” Number

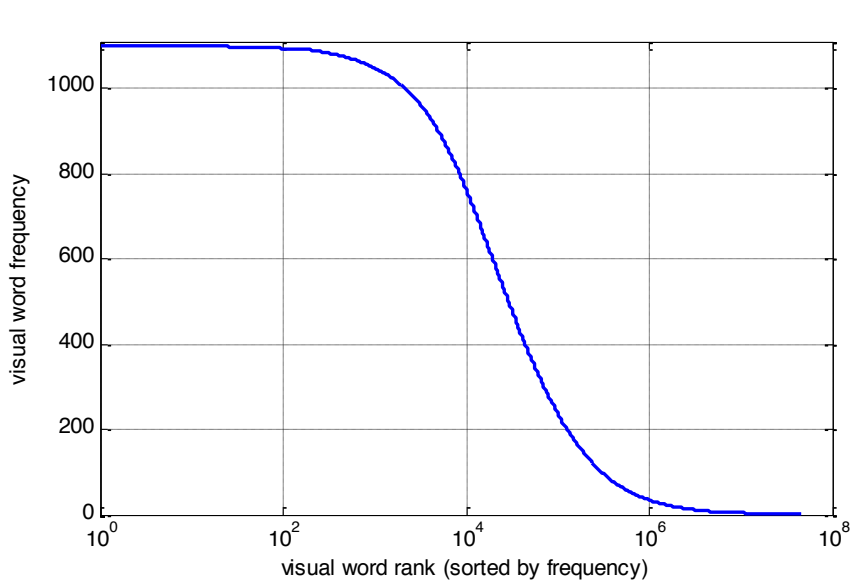
- 32 bits code word address -> 2^{32} (about 4295 million) codewords in total in theory
- Non-empty lists in total 46.5 millions



Stop Word Removal



(a)



(b)

Frequency of code words among one million images (a) before, and (b) after, application of a stop list.

Among the 46.5 M codewords, we further remove those stop words whose occur frequently

Stage III: Codebook-free based CBIR - Scalar Quantization

- SIFT distance distribution of true matches and false matches