A Scalable Approach for Content-Based Image Retrieval in Peer-to-Peer Networks

Lelin Zhang, *Student Member, IEEE,* Zhiyong Wang, *Member, IEEE,* Tao Mei, *Senior Member, IEEE,* and David Dagan Feng, *Fellow, IEEE*

Abstract—Peer-to-peer networking offers a scalable solution for sharing multimedia data across the network. With a large amount of visual data distributed among different nodes, it is an important but challenging issue to perform content-based retrieval in peer-to-peer networks. While most of the existing methods focus on indexing high dimensional visual features and have limitations of scalability, in this paper we propose a scalable approach for content-based image retrieval in peer-to-peer networks by employing the bag-of-visual-words model. Compared with centralized environments, the key challenge is to efficiently obtain a global codebook, as images are distributed across the whole peer-to-peer network. In addition, a peer-to-peer network often evolves dynamically, which makes a static codebook less effective for retrieval tasks. Therefore, we propose a dynamic codebook updating method by optimizing the mutual information between the resultant codebook and relevance information, and the workload balance among nodes that manage different codewords. In order to further improve retrieval performance and reduce network cost, indexing pruning techniques are developed. Our comprehensive experimental results indicate that the proposed approach is scalable in evolving and distributed peer-to-peer networks, while achieving improved retrieval accuracy.

Index Terms—Bag-of-visual-words, content-based image retrieval, peer-to-peer, information maximization, workload balance

1 Introduction

Peer-to-peer (P2P) networks, which are formed by equally privileged nodes connecting to each other in a self-organizing way, have been one of the most important architectures for data sharing. Popular P2P file-sharing networks such as eDonkey[1] count millions of users [1] and tens of millions of files. Unlike webpages which mainly consist of textual documents such as news, blog articles or forum posts, multimedia files play a dominant role in most P2P networks [2]. The ever-growing amount of multimedia data and computational power on P2P networks exposes both the need and potential for large scale multimedia retrieval applications such as content-based image sharing, and copyright infringement detection.

While P2P networks are well known for their efficiency, scalability and robustness on file sharing, providing extended search functionality such as content-based image retrieval (CBIR) faces the following challenges: 1) in contrast to centralized environments, data in P2P networks is distributed among different nodes, thus a CBIR algorithm needs to index and search for images in a distributed manner; 2) unlike distributed servers/clouds, nodes in P2P networks have limited network bandwidth and computational power, thus the algorithm should keep the network cost low and the workload among nodes balanced; and 3) as P2P networks are under constant churn, where nodes join/leave and files publish to/remove from the network, the index needs to be updated dynamically to adapt to such changes.

To support content indexing and avoid message flooding, structured overlay networks such as Distributed Hash Tables (DHTs) [3], [4] are often implemented on top of a physical network. By organizing the nodes in a structured way, messages can be efficiently routed between any pair of nodes, and the index integrity can be maintained during network churn. For the CBIR functionality, most of the existing systems adopt a global feature approach: an image is represented as a high-dimensional feature vector (e.g., color histogram), and the similarity between files is measured using the distance between two feature vectors [5], [6], [7]. Usually, the feature vectors are indexed by a distributed high-dimensional index or Locality Sensitive Hashing (LSH) over the DHT overlay. However, due to the limitation known as "curse of dimensionality", the majority of these solutions have high network costs or serious workload balance issue among nodes when the dimensionality of feature vectors is high.

On the other hand, the bag-of-visual-words (BoVW) model has been successfully utilized for large scale image retrieval [8]. In the BoVW model, each image is represented with a bag of local features, which mimics the bag-of-words (BoW) model where each document is a collection of unordered words.[2] Generally, to employ the BoVW model, the following three steps are required [9], [10]: Firstly, a number of local regions (through image segmentation or uniform image partitioning) or keypoints (through keypoint

- *L. Zhang, Z. Wang, and D. Feng are with the School of Information Technologies, The University of Sydney, NSW 2006, Australia. E-mail: lzha1533@uni.sydney.edu.au; zhiyong.wang@sydney.edu.au; dagan.feng@sydney.edu.au.*
- *T. Mei is with Microsoft Research, Beijing 100080, China. E-mail: tmei@microsoft.com.*

Manuscript received MMM DD, YYYY; revised MMM DD, YYYY.

1. www.emule-project.net

2. As the BoVW model is an analogy to the BoW model, from the retrieval point of view, "term", "visual word" and "codeword"; "codebook" and "vocabulary"; "document", "file" and "image" are analogous concepts. For the convenience of our discussion, if no confusion is caused, we use them interchangeably.

detection algorithms such as Hessian-Affine detector [11]) will be identified from an image and each region or keypoint will be represented with a high dimensional descriptor. In our experiments, the widely used Scale-Invariant Feature Transform (SIFT) descriptor [12] is employed.

Secondly, since the features extracted are in a continuous space, a codebook is generated to quantize the feature vectors into discrete codewords, thus an image can be interpreted as a set of feature codewords. One of the most commonly used quantization schemes is nearest-neighbor quantization (e.g., k-means [9], [10]), where each feature vector is represented by its nearest codeword centroid, and the codebook forms a Voronoi partitioning of the feature space.

Lastly, similar to the BoW model, statistical distributions of the codewords in a given image is utilized to represent the image. In this paper we utilize the well-studied *tf-idf* weighting scheme and cosine distance as the similarity measurement.

To implement these steps in P2P networks, each node firstly performs feature extraction and quantization locally. Therefore, the key challenges lie in the steps performed in a distributed way: the codebook generation/updating and retrieval. As discussed earlier, the algorithms need to have a low and even network cost on all nodes, and adapt to data dynamics. Table 1 shows the estimated per-node computation and network cost of different steps in our experimental system, assuming an average bandwidth and a typical DHT implementation. As shown in the table, the efficiency of a retrieval system mainly depends on the network cost between nodes, rather than the computation cost within a node. Therefore it is essential to minimize the network cost and keep the workload balanced during both codebook updating and retrieval.

For data dynamics, the data in a P2P network is under constant churn. The codebook in such an environment needs to be updated periodically, rather than kept static. At the same time, during the retrieval process, relevance information can be accumulated explicitly by users providing feedback about their query results; or implicitly from the downloading behavior after a query, which can be utilized to improve retrieval performance [15]. With various dynamic data, updating the codebook incrementally and continuously to maintain optimal performance is very challenging.

To address the above challenges, in this paper, we present a novel method to dynamically generate and update a global codebook, which considers both the discriminability and workload balance. While processing queries, each node collects the relevance information and workload data. With the relevance information, we maximize the information provided by the codebook about the retrieval results, thus minimizing the information loss incurred by quantization. With workload data, we aim to achieve a fair workload among nodes, thus avoiding overloading or underloading nodes. Based on these two criteria, the codebook partitioning is updated routinely by splitting/merging codewords, thus allowing the codebook to grow/shrink in accordance to the data distribution. To minimize the cost of codebook updating, the decision whether a codeword should be split/merged is taken by its managing node individually. Finally, the updates are synchronized across the network at the end of each iteration. As a result, the discriminability and workload balance is optimized continuously with the churn of the P2P network.

For the retrieval process, we are able to leverage the existing research on P2P-based text retrieval systems [16], [17], [18], [19], as the BoVW model is an analogy to the BoW model. Specifically, we build two logical indices over DHT: a file index and a codeword index. The file index manages file publishing and removing, while codeword index serves as an inverted index for feature posting lookup. To reduce the query network cost, we applied index pruning techniques to discard postings that are not likely to contribute to top retrieval results.

Overall, the key contributions of our work are:

- It is the first study to investigate scalable CBIR with the BoVW model in P2P networks.
- A novel objective function for codebook optimization in a P2P environment is proposed, which considers both the relevance information and the workload balance simultaneously.
- A distributed codebook updating algorithm based on splitting/merging of individual codewords is proposed, which optimizes the objective function with low updating cost.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 provides an overview of our network structure. Section 4 discusses our proposed codebook generation and updating algorithms. Section 5 discusses the retrieval process. Section 6 presents the experimental results and discussions. Finally, Section 7 concludes this paper.

2 Related Work

2.1 CBIR Feature Representation and Indexing

As mentioned earlier, in order to support CBIR in P2P networks, structured overlay networks are often implemented on top of a physical network. A popular class of overlay networks is Distributed Hash Table (DHT) [3], [4], which builds a hash table globally and stores the entries among the nodes with corresponding hash ID. With the overlay network, each node needs to be able to represent the files with features, and store/retrieve the features to/from the global structure efficiently. Achieving this is very challenging, as we need to align the feature representation and indexing method with the underlying overlay. Generally, existing approaches can be divided into the following two streams.

2.1.1 Global Feature Model

The global feature model represents each image with one high-dimensional feature vector, and measures the similarity between images with the distance between their feature vectors. This model is adopted by many existing P2P CBIR systems [5], [6], [20], [21], [22], [23], [24]. To store and retrieve the feature vectors efficiently, generally two categories of methods can be applied: distributed high-dimensional indexing and Locality-Sensitive Hashing (LSH).

The high-dimensional indexing based approaches store the feature vectors in a data structure, usually a tree or

TABLE 1
Estimated Per-Node Computation and Network Cost Estimation of Different BoVW Steps

Dataset	**Feature Extraction**	**Quantiza-tion**	**Retrieval**				**Codebook Updating**	
			CPU	**DHT**	**Avg. Traffic**	**Index Transfer**	**CPU**	**Avg. Traffic**
UKBench	0.188 s	1.168 s	0.037 s	≈ 1 s	2.975 – 16.175 KB	0.984 – 27.804 s	< 2 s	1.456 – 80.913 KB
Holidays	0.554 s	2.818 s	0.048 s		0.476 – 3.507 KB	0.117 – 17.663 s		1.280 – 35.641 KB
Holidays + Flickr100K		11.220 s	0.376 s		5.519 – 40.660 KB	1.612 – 243.301 s		69.511 – 337.926 KB

*Ranged values indicate the best/worst values obtained with different methods and settings we evaluated. **Feature Extraction**: The average time to extract the SIFT features of an image. **Quantization**: The average time to quantize the SIFT features into codewords. **Retrieval-CPU**: The average time to compute the similarities of a query image. **Retrieval-DHT**: The delay of index lookup in a DHT network [13]. **Retrieval-Avg. Traffic**: The average network traffic of all the nodes involved in a query, assuming a cost of 20 bytes per posting. **Retrieval-Index Transfer**: The average time to transfer all the postings to the query node, assuming a bandwidth of 3.3 Mbps between nodes (2013 Q2 global average [14]). **Codebook Updating-CPU**: the average time to update the codeword on each node. **Codebook Updating-Avg. Traffic**: The average network traffic of all codeword nodes to update the codebook, assuming a cost of 160 bytes per descriptor.

a graph, to achieve effective search space pruning during retrieval. In structured P2P networks, the high-dimensional index is defined in a distributed way over the P2P overlay, where each node manages a part of the index. For example, in M-Chord [20], data structures based on distances to reference points are built to facilitate metric-based similarity search in Chord [3] networks. In [6], a novel naming mechanism and a tree summarization strategy are employed to build a tree structure over a DHT overlay. However, even in a centralized environment, the performance of high-dimensional indexing suffers from the well-known "curse of dimensionality". That is, as the dimensionality of a feature space increases, the performance of the index decreases rapidly, and a search needs to traverse a larger part of the index [25], [26], [27]. In P2P networks, the situation becomes worse by the fact that the index structure is distributed, as we need to take into consideration the cost of data transfer between nodes.

On the other hand, the Locality-Sensitive Hashing (LSH) based approaches use special hash functions that output the same value for similar objects. In P2P networks, these hash functions are usually combined with the hash table interface of DHTs [21], [22], [23], [24], thus similar feature vectors are stored on the same/neighboring nodes to enable efficient similarity searches. To improve the locality of the hash functions, most works compromise the even distribution of hash buckets [28], [29], which translates to an imbalanced workload among nodes in a P2P network. Some works [23], [28], [29] do tackle this issue by learning more independent hash functions, but all of them perform one-time learning without considering the changes of data distribution that is common in P2P networks due to network churn. Even when one can update the hash functions with changing data, implementing it over the DHTs is very challenging. As the data is stored among nodes of corresponding hash ID, a one-bit change of the hash function output will result in large portion of (if not all) data being assigned to a different node, causing heavy network traffic.

We note that the BoVW histogram, which will be discussed later, can also be considered and processed as a high-dimensional global feature. However, as the BoVW histogram can have a dimensionality of thousands or even millions [30], it is beyond the capacity of existing high-dimensional indexing techniques. While LSH can handle higher dimensionality, it requires longer hash codes to maintain the efficiency [28], causing high network cost when implemented over a DHT.

2.1.2 BoVW Model

The bag-of-visual-words (BoVW) model represents each image with a bag of quantized codewords derived from local features, and measures the similarity between images with the BoVW histogram analogous to a bag-of-words (BoW) model of text retrieval [10]. The retrieval process is typically supported by an inverted index. Though we are not aware of any BoVW based P2P CBIR systems, many existing P2P text retrieval systems build a distributed inverted index in a highly efficient manner over DHT, using term ID as key and document ID as value [16], [17], [18], [19].

Generally, there are two strategies to distribute index tuples: document partition (or local indexing), and term partition (or global indexing), both are well exploited in the literature [31], [32], [33]. With document partition, each node manages an index for a subset of documents. A query will be sent to all index nodes, and be answered by combining the lists of candidate documents returned from them. With term partition, each node manages an index for a subset of terms. A query will only be sent to the nodes managing corresponding terms, and answered by combining the inverted list returned from them. Therefore, document partition typically has a higher network cost than term partition, especially when the index has a good term sparsity [32]. This is not a very big issue in shared-memory or distributed servers, but does pose a challenge in P2P networks, as the nodes in P2P networks are loosely coupled and have much lower bandwidth. As a result, term partition is a more popular choice in P2P networks [16], [17], [18], [19].

To further reduce the network cost and tackle the issue of workload balance with term partition, different techniques have been proposed. For BoVW based CBIR in P2P networks, our previous work [34] proposes a codebook re-sampling mechanism to split the overloaded codewords and merge the underloaded codewords to maintain a balanced workload among different codewords. However, it does not take the relevance information into account. In addition, the split/merge is based on random re-sampling, which is heuristic.

Besides P2P networks, a BoVW based CBIR system in distributed servers is proposed in [33], which seems most relevant to our work. It builds an inverted index among distributed servers with term partition. Learning processes are used to first filter the terms to reduce network cost, then distribute the terms into different servers to improve workload balance. Although the objectives are similar, their method is not directly applicable in P2P networks, as their learning method is designed to be an off-line process, which can not deal with data under constant churn. It will incur high network cost if their learning method is performed in an on-line manner to keep up with changing data, as it requires co-occurrence information among all the terms to be collected and analyzed. Our proposed method achieve this in a very different way: we keep the term distribution unchanged, but update the codebook (the way each term is defined) to maintain the performance when data is changed. In this way, nodes managing different terms can adjust the workload individually with a much lower network cost.

2.2 BoVW Codebook Generation

Unlike the BoW model, which has a natural vocabulary, the visual words of the BoVW model are obtained by quantizing the features using the codebook. Unsupervised methods such as k-means and sparse coding [35] aim to minimize the distortion between the original features and the quantized codewords. On the other hand, in [36], a supervised learning process is used to generate a discriminative codebook, where the loss of information provided by the codebook about the training samples is minimized. In [37], the codebook compactness, along with the discriminability is optimized. Alternatively, LSH methods [28] are also exploited for quantization. However, to the best of our knowledge, none of the existing methods is designed for P2P environments. Besides assigning features to codewords, alternative feature encoding approaches have been proposed. For example, VLAD [38] and Fisher Vector [39] represent features with the deviation from the codewords (or a generative model in Fisher Vector), which showed improved performance on many retrieval and classification datasets. Most of the methods need to process the entire data collection in a centralized manner, which is infeasible in P2P networks. In addition, the specific issues for a distributed codebook such as network cost, workload balance and data churn are not well investigated.

Besides the quantization method, the performance of a codebook is also affected by its size. It is reported by many papers that a sophisticated method can be easily outperformed by a larger codebook [8], [30], [36], as more codewords generally leads to finer-grain quantization. However, while a larger codebook yields better performance, it requires more computational resources. In centralized servers or clusters, the size of the codebook is usually predetermined, as available computational resources are fixed. However, in P2P networks, the available resources is under constant change, as peers join/leave the network. A predetermined codebook size is unlikely to produce optimal performance.

Therefore, our proposed codebook learning method takes both codebook discriminability and workload balance into consideration. The discriminability is measured by the mutual information provided by the codebook about user feedback, which is partially inspired by [36]. However, since

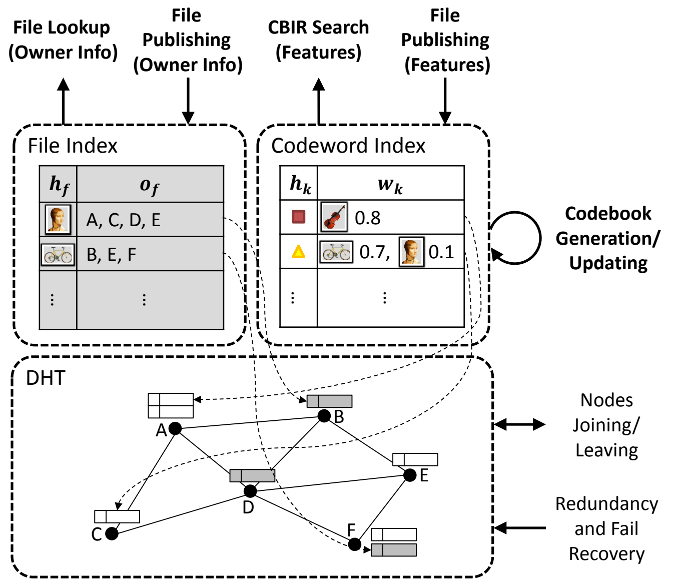


Fig. 1. Illustration of the network overlay structure. The proposed system builds a file index and a codeword index over the DHT overlay. The file index consists of file IDs (h_f) and its corresponding owners (o_f), while the codeword index consists of codeword IDs (h_k) and its corresponding postings (w_k). For example, in the codeword index, the second codeword has two postings: the bike image with a feature posting of 0.7, and the face image 0.1. The entries are distributed to the nodes of the network according to their keys, which are depicted by the dashed arrows from file/codeword index to DHT.

our target application is CBIR and [36] targets classification, the objective function we derived is significantly different. The workload balance is measured by the difference between the current and "ideal" workload for each codeword. To make our codebook adaptive to dynamic P2P environments, the codebook partitioning is optimized by splitting/merging codewords, thereby allowing the codebook to grow/shrink in accordance to the data distribution and available resources. In summary, our algorithm is tailored to deal with distributed and highly dynamic P2P environments.

3 Design Overview

As discussed in Section 2.1, to facilitate the BoVW retrieval process, our system builds inverted indices over the hash table interface of DHT. Before any further discussion, we briefly review DHT. DHT is a class of structured P2P overlay networks that provides $\textsc{Get}(k)$ and $\textsc{Put}(k, v)$ operations similar to a hash table, where k, v are the key and value of a table entry, respectively. While different DHT implementations organize node connections with different topologies, most of them guarantee that a message from any node can reach the corresponding node in $O(\log n)$ hops, where n is the number of nodes. Additionally, DHT handles most issues in node management, including redundancy and failure recovery mechanisms in cases of nodes joining/leaving/failing, and caching and content mirroring for hot spots. Therefore, DHT forms an infrastructure that can be used to build more complex applications.

In order to support various operations of our CBIR system, we build a file index and a codeword index over DHT,

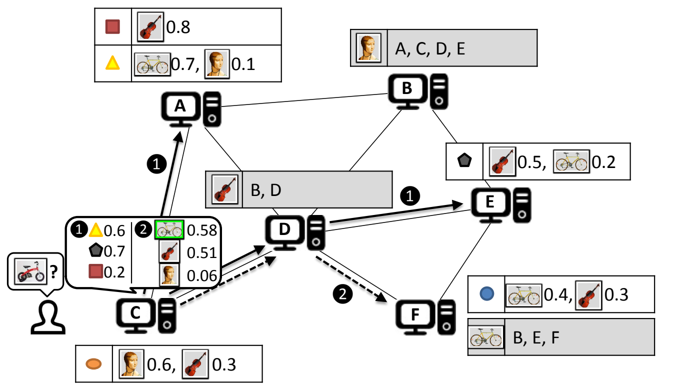


Fig. 2. Illustration of the CBIR process of a query over the DHT overlay network. The network has six nodes $(A, B, \ldots, F)$, and three images (face, yellow bike, and violin). The entries of the file index (marked with gray background) and codeword index (marked with white background) are stored distributedly among different nodes. A CBIR query is answered by first extracting the BoVW representation of the query image locally (list 1 of node C), looking up corresponding postings (solid arrows), computing the similarities and produce the rank list (list 2 of node c), and finally looking up the owners of the relevant image (dashed arrow).

as illustrated in Fig. 1. The file index stores (h_f, o_f) entries with file ID h_f as key, and the file ownership information o_f as value. The codeword index, which stores the postings of each codeword, is added to support the storage and retrieval of BoVW features. It is essentially an inverted index which stores (h_k, w_k) entries with codeword ID h_k as DHT key, and the corresponding postings w_k as value. Each entry is distributed to a node of the network according to its key. For simplicity we refer to a node responsible for an index entry of file f as file node p_f, and the node responsible for an index entry of codeword k as codeword node p_k. The separation of the file and codeword indices is logical, since a node may be responsible for any number of file index entries and/or feature index entries.

All the operations of the CBIR system are translated into lookup or modification of the entries of the file and/or codeword index, which are implemented by GET and PUT operations over the DHT overlay. Details of the operations are listed in the following subsections. Note that node joining/leaving operations are handled by the underlying DHT network and therefore not listed here, and we only need to handle the resultant file publishing/removing operations.

3.1 File Index

3.1.1 File Lookup

Looking up the owners of an exact file is performed with a DHT lookup operation: given a file ID h_f, a list of nodes that has a copy of the file is returned by $\textsc{Get}(h_f)$.

3.1.2 File Publishing/Removing

Publishing a new file is performed by a DHT store operation: the file ID h_f and the list of owner nodes o_f are stored by $\textsc{Put}(h_f, o_f)$. For performance and fault tolerance considerations, such information needs to be reposted periodically, otherwise it would be removed from the owner list o_f. Therefore, removing an entry is achieved by stopping reposting.

3.2 Codeword Index

3.2.1 CBIR Search

The CBIR search is essentially an inverted index lookup in the codeword index. As illustrated in Fig. 2, a user on node C submits a CBIR query with an example image (red bike), which will be answered in three steps: Firstly, the BoVW codewords will be extracted on node C locally. Secondly, the codewords will be looked up in the codeword index with $\textsc{Get}(k)$, where k is the codeword ID (solid arrows). The postings with their corresponding file IDs will be returned by the respective nodes (E and A), which will be used as similarity measurement to produce the ranked results for the user. Finally, the owner information of relevant images can be obtained by the file lookup process described before.

3.2.2 File Publishing/Removing

When a new file is added, besides publishing an entry to the file index with $\textsc{Put}(h_f, o_f)$, the file owner will also extract and quantize the features to form codewords, then put them to the corresponding entries in the codeword index with $\textsc{Put}(h_k, w_k)$. When a file is removed from the file index (with no owner), the corresponding codeword postings will be removed from the codeword index.

3.2.3 Codebook Updating

As to be discussed in Section 4, the global BoVW codebook is updated via splitting and merging codewords. The SPLIT/MERGE operations are essentially publishing/removing entries of the codeword index. For a codeword k stored on node p_k, the SPLIT and MERGE operations are implemented as follows:

SPLIT: To split the codeword k into n codewords, p_k randomly selects $n - 1$ neighboring nodes as new codeword nodes and sends the centroid coordinates to them. Once all the new centroids register themselves as codeword nodes, the descriptor associations of selected nearby partitions will be updated respectively similar to the file posting process.

MERGE: To merge the codeword k, p_k de-registers itself as a codeword node, thus removing k from the global codebook similar to the file removing process. After that, p_k will transfer the descriptors that associate with k to their corresponding new codeword nodes.

3.3 Complexity Analysis

3.3.1 Query Cost

Assume that a P2P network consists of n nodes sharing a total of N images, a codebook K of initial size $|K|$ is utilized for BoVW based retrieval, and each image has an average of c codewords. Our system completes a query in the following steps: 1) feature extraction; 2) quantization; 3a) sending posting lookup message; 3b) receiving postings; and 4) aggregating postings and producing the rank list. We only discuss steps 2–4 as feature extraction time is not affected by our system configuration.

Step 2: When a user submits a query image, the local features are extracted and quantized into c codewords. For the exact nearest neighbor quantization we used in the experiment, the quantization takes $O(c|K|)$ time.

Step 3a: To lookup the postings, c lookup messages are sent in parallel to the corresponding nodes. The bandwidth for sending the lookup messages is $O(c)$, and it takes $O(\log n)$ hops to arrive at the corresponding codeword nodes, thanks to the underlying DHT overlay.

Step 3b: Once the lookup message is received, the codeword nodes will directly send the postings back to the query node, since the source of the query is known. Therefore, the time for this step is determined by the total number of postings returned to the query node, which is $O(cs)$, where $s = cN/|K|$ is the average number of postings stored in the index for each codeword.

Step 4: The aggregation needs to process all the postings, so the complexity is the same as step 3b ($O(cs)$).

Therefore, the total retrieval time complexity is $O(c|K| + c + \log n + c^2 N/|K|)$. The number of codewords per image c is bounded by the number of local features per image, which can be seen as a constant, and the total number of images N is linearly correlated to the total number of nodes n. Hence the time complexity is reduced to $O(|K| + \log n + n/|K|)$.

In our system, we let the codebook size $|K|$ grow as more nodes join the network, therefore $n/|K|$ is a constant and the time complexity becomes $O(|K| + \log n)$, which is very scalable as the network grows. The bottleneck term $|K|$ lies on the quantization step, which can be further improved if we use approximate nearest neighbor instead.

Also note that the retrieval scope—the number of nodes visited during a query, has the complexity of $O(c \log n)$. As c is bounded, this is reduced to $O(\log n)$, which is also scalable as the network grows. Therefore, our proposed retrieval approach is scalable in terms of both query cost and scope.

3.3.2 *Codebook Updating Cost*

For codebook generation and update, each iteration consists of three steps: 1) determine the update operation (split, merge or no change) for each codeword; 2) for split and merge, transfer the postings to/from neighbor nodes; and 3) synchronize the new set of codewords across the network.

Step 1: For each codeword, all the postings need to be scanned once for all the candidate update operations and the best one needs to be chosen. Therefore the cost is $O(sm)$, where $s = cN/|K|$ as before, and m is the number of candidate update operations. As we usually have a maximum limit of split sub-partitioning for each iteration, m can be regarded as a constant. Therefore the cost is reduced to $O(cN/|K|)$.

Step 2: For the split and merge of one codeword, the cost to transfer the postings is $O(s) = O(cN/|K|)$. For the whole network, the cost is $O(ps) = O(pcN/|K|)$, where p is the codebook change ratio (the split/merge codewords vs. total number of codewords).

Step 3: The cost of synchronizing the new codebook is simply $O(p|K|)$, since each node needs to obtain one copy of the updated codewords.

Therefore, the codebook update complexity is $O(cN/|K| + pcN/|K| + p|K|)$. Applying the same treatment for c and N as the query complexity, we get $O(n/|K| + pn/|K| + p|K|)$. Similarly, when the codebook size $|K|$ grows with the network size n, the cost is reduced to $O(p|K|)$, which is scalable to the network growth. Therefore, it is important to maintain a dynamic codebook in P2P environments. Moreover, with $O(p|K|)$ complexity, it is very essential for a codebook update method to minimize the codebook change ratio p. As our experimental result shows, the proposed method has lower codebook change ratio than other methods.

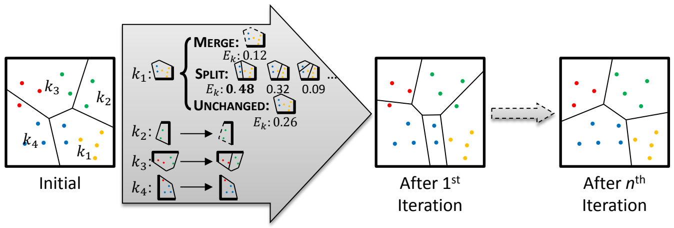


Fig. 3. Illustration of the codebook updating process. Descriptors (dots) relevant to different queries are marked with different colors. Within one iteration, each codeword decides whether it should be split/merged/unchanged based on the resultant objective function values (E_k as to be discussed in Section 4.2, Eq. (11)). In the first iteration, k_1 decides to split into 2 partitions, while k_2 decides to merge. The results are broadcast throughout the network to form a new global codebook. The iterative updating process gradually optimizes the codeword partitioning to form an improved codebook.

4 Codebook Generation and Updating

As illustrated in Fig. 3, our codebook updating algorithm runs iteratively. During an updating iteration, each codeword node p_k decides whether its codeword k should be split/merged/unchanged based on the relevance information collected from past queries, and the current workload. After each iteration, the centroid coordinates and the codeword statistics needed for similarity measurement (e.g., document frequencies) will be broadcasted throughout the network, so that all the nodes in the network can have the same codebook. The iterative process runs continuously in order to maintain an updated codebook during data churn. The frequency of update iterations is determined by the degree of churn. As shown by the experiments in Section 6.3, a very low update frequency (once a few hour) is enough to maintain the performance.

The rest of this section is organized as follows: Sections 4.1 and 4.2 present the two parts of our objective function: mutual information and workload balance, and the two parts are combined in Section 4.3. Section 4.4 introduces the decision-making process of codeword optimization.

4.1 Codebook Information Maximization

In terms of information maximization, we aim to find a partitioning of the feature space such that partitions/codewords are correlated to the collected relevance information.

To achieve this, we model the BoVW based CBIR process with information theory: given the two sets of descriptors Q and X—extracted from query image and candidate images respectively, the objective is to find out the subset of descriptors in X that comes from the images related to Q. In other words, given Q, for each descriptor $x \in X$, one needs to determine the relevance of x, or whether x comes from a relevant image. Denote the relevance information as Y, the amount of information provided by the descriptors can be represented by the conditional mutual information of X and Y under all $Q \in \mathbb{Q}$: $I(X;Y|\mathbb{Q})$. Note that we use a codebook K to quantize the descriptors, and use the resultant codewords to perform the retrieval, the amount of information provided by the codewords can be denoted as $I(K;Y|\mathbb{Q})$.

Naturally, an optimal codebook is the one that minimizes the information loss incurred by the quantization process:

$$\arg\min_K I(X;Y|\mathbb{Q}) - I(K;Y|\mathbb{Q}). \tag{1}$$

Since the distribution of X and Y are both fixed when Q is given, $I(X;Y|\mathbb{Q})$ is fixed. Minimizing Eq. (1) is equivalent to:

$$\arg\max_K I(K;Y|\mathbb{Q}). \tag{2}$$

That is, we seek a quantization method that provides maximum amount of information about the relevance information over all queries.

With our quantizer codebook K which partitions the feature space into k codewords, we express the mutual information by the Kullback-Leibler divergence [40]:

$$\begin{aligned} & I(K;Y|\mathbb{Q}) \\ &= \sum_{Q\in\mathbb{Q}} P(Q)\{D_{KL}[P(K,Y|Q)\|P(K|Q)P(Y|Q))]\} \\ &= \sum_{Q\in\mathbb{Q}}\sum_{k\in K}\sum_{y\in Y} P(Q)P(k|Q)P(y|Q,k)\log\frac{P(y|Q,k)}{P(y|Q)}. \end{aligned} \tag{3}$$

The relevant information Y consists of two parts: the descriptors from a relevant ($y = r$) or irrelevant ($y = \bar{r}$) image. However, most information retrieval algorithms only consider the relevant part of Y, because the inclusion of the irrelevant part leads to a somewhat odd prediction that the presence of a query term is against retrieval [41]. In the experiment, we follow this convention and let $Y = \{r\}$.

The probabilities of Eq. (3) can be derived empirically from index statistics and relevance information.

$P(Q)$, the probability of query Q, is simply:

$$\hat{P}(Q) = \frac{1}{|\mathbb{Q}|}. \tag{4}$$

$P(k|Q)$, the probability of a retrieved descriptor comes from partition k under query Q, is given by:

$$\hat{P}(k|Q) = \max_{q\in Q} w_{k,q}\frac{\sum_{x\in X} w_{k,x}}{\sum_{x\in X}\sum_{k\in K} w_{k,x}}, \tag{5}$$

where $w_{k,d}$ is the assignment weight of a descriptor d to partition k, for the hard-assignment codebook we used, $w_{k,d} = \{1 \text{ if } d \text{ is quantized as codeword } k, 0 \text{ otherwise}\}$. The left part—the maximum query assignment weight in k, considers whether the codeword k will be retrieved by a query descriptor $q \in Q$. For hard-assignment codebook, $\max_{q\in Q} w_{k,q} = 1$ as long as at least one query descriptor q is quantized as k, therefore codeword k will be retrieved. The right part—the ratio of total weight between k and all partitions, considers if codeword k is to be retrieved, the likelihood of a candidate descriptor d belonging to k.

$P(y|Q,k)$, the probability of getting a relevant ($y = r$) or irrelevant ($y = \bar{r}$) descriptor in partition k given query Q, is given by the portion of relevant/irrelevant descriptors within partition k. Likewise, $P(y|Q)$, the probability of getting a relevant/irrelevant descriptor in all partitions given query Q, is given by the portion of relevant/irrelevant descriptors in all partitions. Without loss of generality, we consider the case of $y = r$:

$$\begin{aligned} \hat{P}(r|Q,k) &= \frac{\sum_{x\in X}\hat{P}(r|x,Q)w_{k,x}}{\sum_{x\in X} w_{k,x}}, \\ \hat{P}(r|Q) &= \frac{\sum_{x\in X}\sum_{k\in K}\hat{P}(r|x,Q)w_{k,x}}{\sum_{x\in X}\sum_{k\in K} w_{k,x}}, \end{aligned} \tag{6}$$

where $\hat{P}(r|x,Q)$, the relevance of x and Q, is given by:

$$\hat{P}(r|x,Q) = \begin{cases} 1 & \text{if } x \text{ is relevant to } Q; \\ 0 & \text{if } x \text{ is irrelevant to } Q; \\ P_u & \text{if relevance is unknown.} \end{cases} \tag{7}$$

Normally we have $P_u = N_r/N$, where N_r is the average number of relevant images given a query, and N is the total number of images in the dataset.

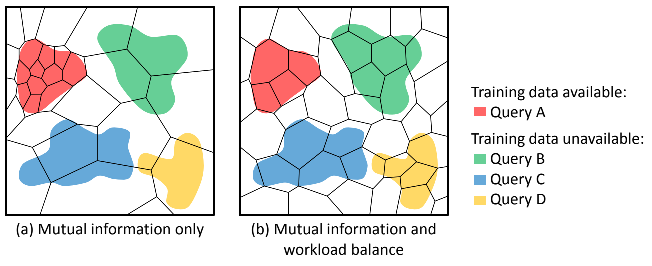

Training data available:
🟥 Query A
Training data unavailable:
🟩 Query B
🟦 Query C
🟨 Query D
(a) Mutual information only
(b) Mutual information and workload balance

Fig. 4. Illustration of the bias towards partial training data. Assume we only have relevance information for query A, if we only consider mutual information of available training data, the resultant partitioning will be similar to (a), which divides the area of query A with small partitions, but leaves the rest areas (queries B, C and D) coarsely partitioned. If we consider both mutual information and workload balance, the resultant partitioning will be similar to (b), which is likely to get better results for queries B, C and D due to its finer-grain partition in these areas.

4.2 Workload Balance

For workload balance, we aim to partition the feature space evenly and accommodate the computational capacity of each nodes, so that no nodes would be overloaded or underloaded.

While achieving good discriminability is important, a codebook used in P2P networks must also produce fair workload for each node. Unfortunately, in real world scenarios where relevance information is incomplete, the information maximization process often produces imbalanced partitions, as it has strong bias towards available training data. As illustrated in Fig. 4, the resultant codebook tends to use more fine-grained partitions to represent the areas where relevance information is available, but leaves the rest of the areas coarsely partitioned. This not only compromises the retrieval performance of unknown queries, but also produces imbalanced workload: the nodes managing large partitions will be overloaded and the small partitions will be underloaded.

To rectify this problem, for each k we define a workload factor to measure the difference between the current workload s_k and the target workload S_k as:

$$F(k) = W_0 - |\log\frac{s_k}{S_k}|, \tag{8}$$

where W_0 is an offset constant. The function peaks at $s_k = S_k$, and penalizes the case of overload ($s_k > S_k$) and underload ($s_k < S_k$). There are two reasons to measure the difference by logarithmic function: 1) we use split/merge operations to adjust the partitioning, and the sizes of the partitions before and after split/merge exhibit exponential relation; 2) it aligns well with the logarithmic form of mutual information, making the combination of the two criteria simpler.

In many applications, instead of defining target load S_k, it's easier to define the lower bound S_m and upper bound S_M for a fair workload, where $F(k) \geq 0$ when $s_k \in [S_m, S_M]$. In this case S_k can be derived as the geometric mean of S_m and S_M, which yields:

$$F(k) = \frac{1}{2}\log\frac{S_M}{S_m} - 2|\log\frac{s_k}{S_m S_M}|. \quad (9)$$

The average workload difference for all codewords in all queries is given by:

$$F(K) = \sum_{k \in K} \frac{s_k}{N} F(k). \quad (10)$$

4.3 Combining Information Maximization and Workload Balance

We take both the mutual information and the workload balance into account in the objective function as:

$$E_K = I(K; Y|\mathbb{Q}) + \alpha F(K), \quad (11)$$

where α is the relative weighting parameter.

For the simplicity of our discussion, we define $I_K = I(K; Y|\mathbb{Q})$, $F_K = F(K)$, and E_k, I_k, F_k as the respective portion of E_K, I_K, F_K for the partition/sub-partitions of codeword k.

The optimal value of α is application dependent. However, in the case that the relevance information is incomplete, we have $\alpha < N_f/N$, where N_f/N is the portion of the images that bear relevance information. This is because for those images without relevance information, one has to assume a relevance of P_u, as in Eq. (7), thus they make no contribution to I_K. Setting $\alpha < N_f/N$ will balance out this portion of unknown information.

4.4 Codeword Optimization Algorithm

We optimize E_K by finding a suitable partition granularity and good centroid positions. The learning algorithm adjusts the partitioning by splitting/merging the partitions iteratively. Since the codeword index and relevance information is managed by codeword nodes p_k, the decision to split/merge a codeword k is made by p_k individually based on its own data.

Generally, for a partition k, one of three cases applies:

1) **SPLIT**: The size of k is large enough for sub-partitioning, and it is possible to get a good sub-partitioning based on available data. In this case, a SPLIT operation is performed: the new centroids of the sub-partitions of k are published to the codebook, thus splitting k into a few smaller partitions.
2) **MERGE**: The current partition performs badly (has low E_k), but the size of k is not large enough for a good sub-partitioning. In this case, a MERGE operation is performed: k is removed from the codebook, and its data is taken over by its neighbors.
3) **UNCHANGED**: The current partition performs well, and one cannot get a better partitioning with either SPLIT or MERGE. In this case, k remains unchanged.

We first try to SPLIT k to see if a sub-partitioning increases E_k. To get a discriminative yet compact sub-partitioning, we first generate an over-complete set of n candidate centroids C_n, then select the subset from C_n which maximizes E_k. The selection process is similar to feature selection [42], and generally there are two greedy selection schemes: backward elimination and forward selection. Starting from C_n, the backward elimination tries to remove one centroid at a time from the current candidate set C_i. Every centroid in C_i is considered for removal, and the configuration with maximum value of E_k is selected as C_{i-1}. Finally, from the sets $C_1, C_2, \ldots, C_n$, the set with maximum value of E_k is selected as the final partitioning. The forward selection follows a similar process, except it starts from $C_1 = \{c_k\}$ (the centroid of k), and tries to add one centroid at a time to construct $C_1, C_2, \ldots, C_n$. In the experiments we adopt the backward elimination scheme, as it yields slightly better results in our preliminary experiments. If the centroid selection process finds a better sub-partitioning than k itself, the SPLIT operation is performed.

If SPLIT fails to yield a higher E_k, the next step is to determine if we want to MERGE k, or keep it unchanged. This is done by setting a threshold θ_E. A codeword partition k is considered to be inefficient if it has $E_k < \theta_E$, and a MERGE operation is performed. In most cases, the size of a merged partition is small, as it has little or no room for a good sub-partitioning by the SPLIT operation tried before. Therefore, the merging cost is minimized. On the other hand, a partition k is considered to be efficient enough if it has $E_k \geq \theta_E$, and will be kept unchanged.

The threshold θ_E defines a standard for an "efficient" codeword. In our experiments we set $\theta_E = 0$, that is, a codeword should provide enough mutual information to justify the extra or idle workload.

5 BoVW based Retrieval Process

As mentioned in Section 3.2, when the codebook is ready, for a given query, the retrieval process essentially consists of three steps: extracting visual features and obtaining BoVW based representation for the query, retrieving the postings via DHT lookup, and measuring the similarity between the query and candidate images.

In large scale BoW based retrieval systems, index pruning has been used to reduce the retrieval cost. Its basic idea is to identify and discard the postings which are not likely to contribute to top results. Most existing index pruning techniques discard terms based on *tf-idf* postings [43]. In the experiments, two threshold based pruning techniques similar to [34] are implemented:

Reducing the query terms: we apply a threshold θ_Q on the posting $w_{Q,k}$ of query image Q, so that only the terms satisfying $w_{Q,k} > \theta_Q$ will be sent.

Reducing the answer terms: we apply a threshold θ_A on the similarity scores of the candidate postings, so that only the postings satisfying $w_{Q,k}w_{A,k} > \theta_A$ will be sent, where $w_{A,k}$ is the posting of a candidate image A.

6 Evaluation

6.1 Experimental Settings

The experiments are conducted on two publicly available datasets:

UKBench [30]: A benchmark dataset for object recognition. It contains 10,200 images with 4 images per object in different conditions. In our experiments, an image is considered relevant to the query image if both of them come from the same object. The SIFT descriptors as in [44] are used as local descriptors.

Holidays [45]: A benchmark dataset for image retrieval. It contains 1,491 images with 500 queries and 991 corresponding relevant images. The number of relevant images for each query varies from 1 to 11. The SIFT descriptors coming with the dataset are used as local descriptors.

Additional datasets are used to facilitate the experiments:

Flickr100K [45]: For the distractors for large scale experiments, we use the first 100,000 images from the Flickr1M, where images are downloaded from Flickr.

Flickr60K [45]: For the source of initial codebook, we use an independent dataset Flickr60K, which contains 67,714 images downloaded from Flickr.

We compare the proposed P2P codebook learning method (PCL) with codebook re-sampling (RS) [34] and k-means (KM) clustering under different settings. For PCL, we use the ground truth relevance information from the top 10 results for training.

The codebook re-sampling (RS) updates the codebook using split/merge operations similar to the proposed method. However, the decisions to split/merge are based on the size of codeword:

$$\begin{cases} \textsc{Split}(k) & \text{if } |k| > \theta_M; \\ \textsc{Merge}(k) & \text{if } |k| < \theta_m, \end{cases} \tag{12}$$

where θ_M and θ_m are the workload thresholds for split and merge operations, respectively.

The k-means clustering (KM) updates the codebook by moving the codeword centroids to the cluster means. In the experiments, we implement a distributed version of k-means algorithm. During an iteration, each codeword computes the new centroid from the mean of its descriptors. The new centroids are synchronized across the network to form the codebook for the next iteration.

In order to achieve fair comparisons, we try to set the parameters of different methods to produce codebooks with sizes $k \approx 20,000$, which is a typical number in reported BoVW based CBIR systems [8], [30]. For large scale Holidays + Flickr100K, the codebook sizes are set to $k \approx 100,000$. In other words, we aim at a target workload of $S_k = |X|/k$, where $|X|$ is the total number of descriptors of all candidate images. For PCL, we set the workload lower and upper bounds as $S_m = S_M = S_k$ for Eq. (9) when relevance information is available, and $S_m = 0.5S_k$, $S_M = 2S_k$ when relevance information is missing. For re-sampling, we set the workload thresholds as $\theta_M = 1.5S_k$ and $\theta_m = 0.5S_k$. For k-means, the codebook size is fixed on k.

As discussed in Section 3, both the CBIR search and codebook generation/updating take place on the codeword index. Therefore, we evaluate the proposed system with a multi-threaded program that simulates the codeword index, where the updating process of each codeword node is executed in an individual thread. Although as shown earlier in Table 1, the computation cost is small for individual codeword nodes (e.g., an update iteration takes less than 2 seconds), simulating a large number of codeword nodes still takes considerable amount of CPU time and memory. For the large scale experiment (Holidays + Flickr100K, with 100,000 codeword nodes), it takes more than 1,000 hours of running time, and 380GB of memory.

In correspondence to the challenges in P2P environments discussed in Section 1, we evaluate the following 4 properties of the codebooks in the experiment:

Retrieval accuracy: We follow the recommended evaluation protocols of the datasets to measure the retrieval accuracy. For both datasets, we report the Mean Average Precision (MAP), which is the average area under the precision-recall curve for all the queries; and R-Precision (RP), which is the average precision of top R results, where R is the number of relevant images. For the UKBench dataset, we also report the Kentucky Score (KS) (the average number of positive images in top 4 results, which is essentially RP * 4) used by the dataset authors.

Workload balance: The workload balance is measured by the Gini coefficient [46] among sizes of codewords, where a coefficient of 0 expresses perfect workload balance (all codewords have the same number of descriptors), and a coefficient of 1 expresses maximum workload imbalance (one codeword has all the descriptors).

Updating cost: Based on the discussion in Section 3.3, we measure the updating cost by the codebook change ratio $p = N_C/|K|$, where N_C is the number of codewords changed (either split, merged, or moved) in this iteration, and $|K|$ is the codebook size (total number of codewords).

Query cost: The query cost is measured by the average number of retrieved postings for all the queries. The number determines the data volume that a query node will receive upon a query, which contributes most to the retrieval time.

We compare the performance of different codebooks in 3 scenarios: static environment, dynamic environment, and retrieval with index pruning. The detailed settings for different scenarios are discussed in their corresponding sections.

6.2 Static Environment

6.2.1 Standard Data

We first compare the performance of the proposed PCL method with RS and KM under a static environment, where data do not change. The experiments are conducted using the whole dataset of UKBench and Holidays. To enforce data independence upon initialization, we use the codebook with 20,000 codewords obtained from Flickr60K as the initial codebook. The codebook is then updated with PCL, RS, and KM methods for 10 iterations.

For the UKBench dataset, we report the performance of 3 different settings for PCL: relevance information of top 10 results with $\alpha = \{0.0001, 0.0002\}$, and no relevance

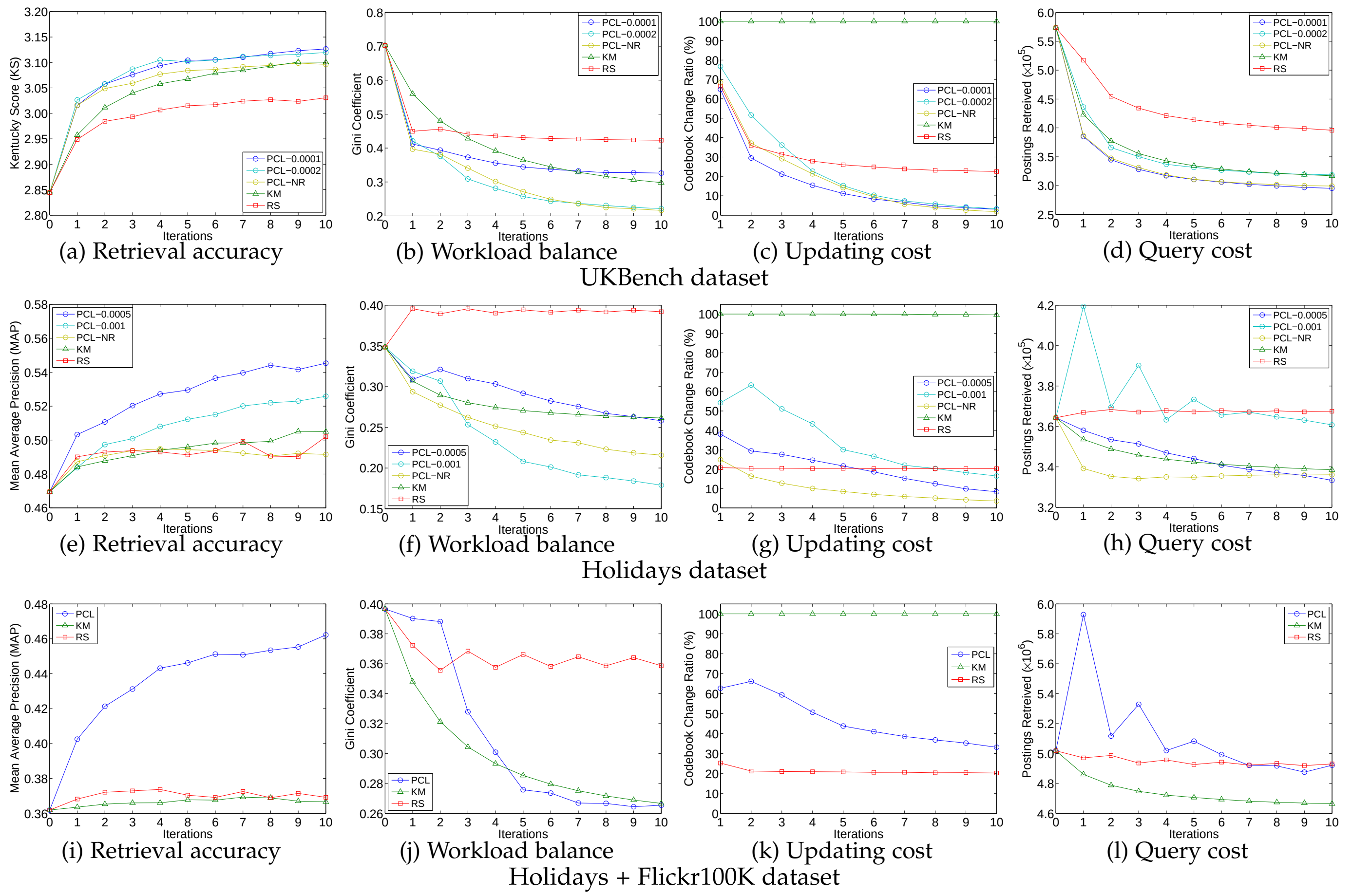


Fig. 5. Performance of the proposed PCL method (with different α values) compared with re-sampling (RS) and k-means (KM) on the UKBench ((a) – (d)), Holidays ((e) – (h)), and Holidays + Flickr100K ((i) – (l)) datasets in a static environment.

information (NR). For the Holidays dataset, the 3 different settings for PCL are $\alpha = \{0.0005, 0.001\}$, and NR.

The results are summarized in Fig. 5 (a) – (h) and Table 2 (a), (b). In terms of retrieval accuracy, all methods achieved significant improvement in the learning process. With relevance information, PCL consistently offers best accuracy, and KM generally performs slightly better than PCL-NR and RS.

On workload balance, all methods produce Gini coefficients < 0.43, which is a strong indicator of a fair workload [23]. PCL generally performs better than or comparable to KM, with the Gini coefficients falling in the range of [0.17, 0.33] at the 10th iteration. RS has a more imbalanced workload, where the Gini coefficients end up as around 0.4 for both datasets.

For the updating cost, PCL incurs minimal change after the first few iterations. The codebooks produced by PCL are very stable at the 10th iteration (except PCL-0.001 for Holidays), with change ratios $< 10\%$. Following is the RS with change ratio stable at around 20%. In contrast, the change ratio of KM remains at the 100% level, i.e., all codeword centroids are moved in each iteration. Generally, KM converges much slower than PCL and RS, and the resultant high updating cost makes it unfavorable for P2P environments.

For the query cost, both PCL (except PCL-0.001 for Holidays) and KM produce similar query costs, reducing the cost by 44.4% – 48.5% for UKBench, and 7.1% – 8.5% for Holidays during the updating process. In contrast, RS produces higher query cost, yielding a cost reduction of 31.0% for UKBench, and an increase of 0.9% for Holidays.

Comparing the results of PCL, we can see that the relative weighting value α is correlated to all performance perspectives. A larger α value puts more emphasis on workload balance, therefore the workload is more balanced, but the updating cost may be higher as we try to balance the partitions vigorously. On the other hand, a smaller α value puts more emphasis on relevance, therefore the retrieval accuracy is higher. In addition, the query cost is usually smaller, since the codewords are more discriminative. Generally, in order to obtain best retrieval accuracy and efficiency, we should aim at a small α value, as long as the workload imbalance problem remains manageable.

The performance on two datasets exhibit different characteristics. Most notably, the initial workload is much more balanced on Holidays than UKBench (Gini coefficient 0.349 vs. 0.702), leaving a smaller margin of improvement for the codebook updating algorithms. This indicates that the initial Flickr60K codebook is a better fit for the Holidays dataset, which is not surprising, considered they are both collected from Flickr. As the hot spots are smoothed out as the workload becomes more balanced, a greater query cost reduction is achieved on UKBench.

TABLE 2
Comparison of Initial and Final Performance between PCL (with Different α Values), RS and KM on the UKBench and Holidays Datasets in a Static Environment

(a) UKBench Dataset

Method	MAP	KS (RP)	Gini	Change %	# of Postings
Initial	0.750	2.844 (0.711)	0.702	n/a	573,453
PCL-0.0001	**0.816**	**3.127 (0.782)**	0.326	3.02%	**(-48.5%) 295,382**
PCL-0.0002	0.815	3.120 (0.780)	0.222	3.29%	(-44.4%) 318,562
PCL-NR	0.811	3.096 (0.774)	**0.216**	**1.80%**	(-47.8%) 299,216
KM	0.814	3.100 (0.775)	0.298	100.00%	(-44.7%) 316,989
RS	0.793	3.031 (0.758)	0.423	22.50%	(-31.0%) 395,825

(b) Holidays Dataset

Method	MAP	RP	Gini	Change %	# of Postings
Initial	0.470	0.429	0.349	n/a	364,305
PCL-0.0005	**0.545**	**0.512**	0.258	8.35%	**(-8.5%) 333,348**
PCL-0.001	0.526	0.492	**0.179**	16.48%	(-1.0%) 360,789
PCL-NR	0.492	0.455	0.216	**3.59%**	(-7.7%) 336,102
KM	0.505	0.466	0.262	99.58%	(-7.1%) 338,529
RS	0.502	0.466	0.392	20.28%	(+0.9%) 367,545

(c) Holidays + Flickr100K Dataset

Method	MAP	RP	Gini	Change %	# of Postings
Initial	0.362	0.335	0.397	n/a	5,018,077
PCL	**0.462**	**0.446**	**0.267**	33.11%	(-1.9%) 4,921,649
KM	0.367	0.336	0.275	100.00%	**(-7.1%) 4,663,822**
RS	0.369	0.330	0.365	**20.22%**	(-1.7%) 4,930,297

6.2.2 *Large Scale Data*

To evaluate the performance with large scale noisy data, we combined the Holidays dataset with the distractors of Flickr100K to form a dataset of 101,491 images. We used a codebook with 100,000 codewords obtained from Flickr60K as the initial codebook, and update the codebook with different methods for 10 iterations. For PCL, we set $\alpha = 0.00001$.

The results are summarized in Fig. 5 (i) – (l) and Table 2 (c). In terms of retrieval accuracy, the proposed PCL method beats KM and RS by a much larger margin compared to results of smaller scale data. This shows the importance of utilizing relevance information when the dataset becomes bigger and noisier. At the same time, PCL also achieved best workload balance and comparable query cost. The increased codebook updating cost is caused by the vigorous setting of α value to optimize workload balance, which can be reduced by reducing the α value.

6.3 Dynamic Environment

To evaluate the performance of the proposed PCL method under network churn, we perform experiments in an iterative setting similar to the static environment. Instead of keeping things unchanged in-between iterations, two types of churns are simulated: data churn and node churn. For data churn, we split the dataset into two halves, using one half (the current set) for codebook updating and evaluation, and the other half as reserve. We swap a random portion (p) of data between the current and reserve set to simulate the change of data. For node churn, we force a random portion (p) of codewords to hand over its postings to neighbors to simulate the drop out of codeword nodes. The churn level p indicates the change rate between two update iterations. In the experiments, we set $p = \{10\%, 50\%\}$ to simulate P2P networks with moderate and extreme churn levels. We use the same 20K codebook obtained from Flickr60K as the initial codebook, and update the codebook with PCL, KM and RS for 15 iterations. For PCL, the α value is experimentally set at 0.0004 for both UKBench and Holidays datasets.

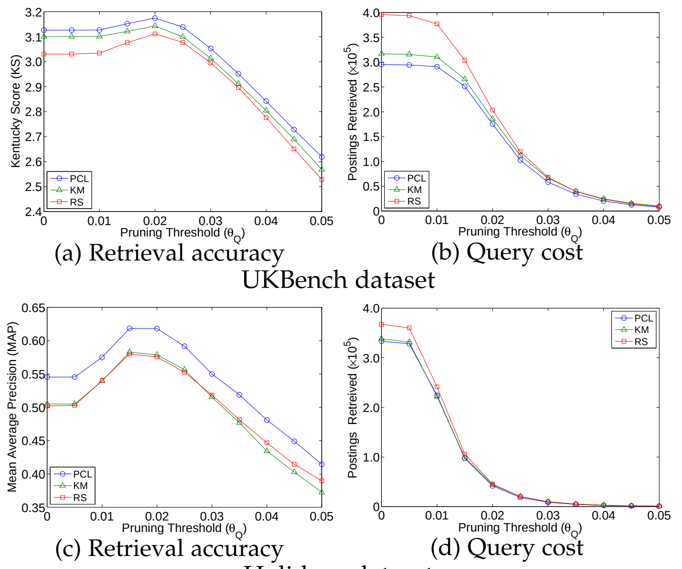
The results are summarized in Fig. 6 and Table 3. Overall, PCL outperforms other methods in terms of workload balance, updating cost and query cost, and equally best with (if not better than) KM on retrieval accuracy. While KM achieved good retrieval accuracy, it has the highest updating cost and worst workload balance. RS achieved reasonable performance on 10% churn level, but failed to keep up with the rapid changes on 50% churn level, as indicated by the dramatic increase of updating cost and query cost.

Comparing the updating costs of PCL and RS in different environments, we can see a correlation between the updating cost and churn level. Such an adaptive nature is essential to minimize the updating cost in the ever-changing P2P networks, as we only update the codewords when necessary. For a real world scenario, it is estimated that the daily node population change rate is about 10% to 15% (in KAD network [1]). Therefore, a low iteration frequency (once a few hour) is sufficient to maintain the performance.

6.4 Index Pruning

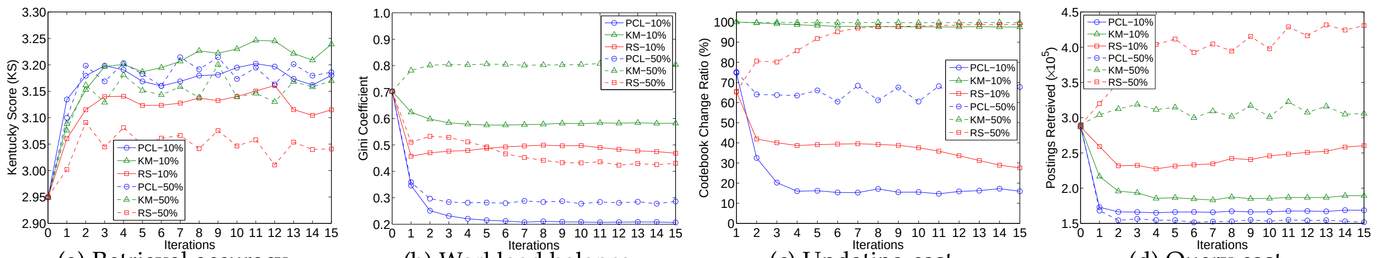
Based on the codebooks obtained after 10 updating iterations in Section 6.2, we utilize the index pruning policies mentioned in Section 5. The θ_Q is set to 11 values ranging from 0 to 0.05, and their corresponding θ_A is set as $\theta_A = {\theta_Q}^2$.

The results are summarized in Fig. 7. The results of both datasets show a similar trend. As the pruning threshold



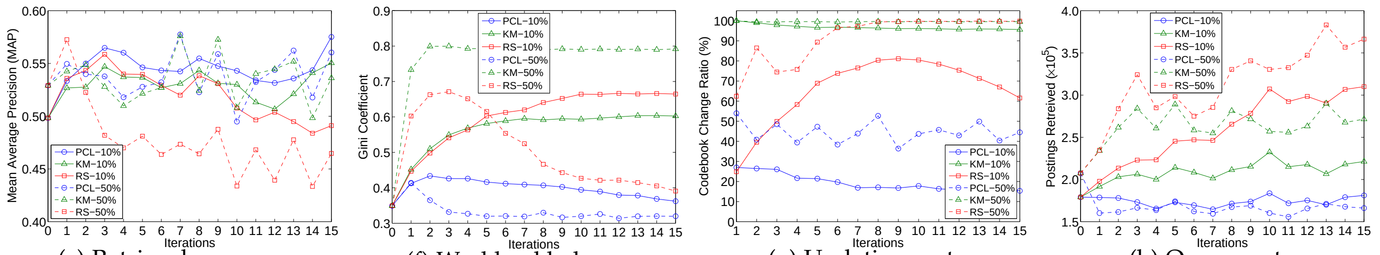
(a) Retrieval accuracy (b) Query cost
UKBench dataset
(c) Retrieval accuracy (d) Query cost
Holidays dataset

Fig. 7. The impact of different index pruning settings on different term weighting schemes on the UKBench ((a) and (b)) and Holidays ((c) and (d)) datasets.



(a) Retrieval accuracy (b) Workload balance (c) Updating cost (d) Query cost

UKBench dataset



(e) Retrieval accuracy (f) Workload balance (g) Updating cost (h) Query cost

Holidays dataset

Fig. 6. Performance of the proposed PCL method compared with k-means (KM) and re-sampling (RS) on the UKBench ((a) – (d)) and Holidays ((e) – (h)) datasets in dynamic environments with different churn levels (10% and 50%).

TABLE 3
Comparison of Initial and Final Performance between PCL and RS on the UKBench and Holidays Datasets in Dynamic Environments with Different Churn Levels

(a) UKBench Dataset

Churn Level: 10%

Method	MAP	KS (RP)	Gini	Change %	# of Postings
Initial	0.776	2.949 (0.738)	0.703	n/a	287,122
PCL	0.831	3.180 (0.795)	**0.207**	**15.95%**	**(-41.3%) 168,594**
KM	**0.845**	**3.239 (0.810)**	0.582	97.48%	(-34.1%) 189,322
RS	0.815	3.115 (0.779)	0.468	27.56%	(-9.4%) 260,166

Churn Level: 50%

Method	MAP	KS (RP)	Gini	Change %	# of Postings
Initial	0.775	2.950 (0.738)	0.703	n/a	289,575
PCL	**0.834**	**3.186 (0.797)**	**0.286**	**67.73%**	**(-47.6%) 151,675**
KM	0.828	3.170 (0.793)	0.804	99.68%	(+5.6%) 305,923
RS	0.800	3.041 (0.760)	0.430	98.88%	(+48.7%) 430,641

(b) Holidays Dataset

Churn Level: 10%

Method	MAP	RP	Gini	Change %	# of Postings
Initial	0.498	0.456	0.349	n/a	179,137
PCL	**0.575**	**0.542**	**0.362**	**15.36%**	**(+1.1%) 181,174**
KM	0.550	0.505	0.603	95.63%	(+23.6%) 221,347
RS	0.491	0.461	0.665	61.59%	(+73.0%) 309,951

Churn Level: 50%

Method	MAP	RP	Gini	Change %	# of Postings
Initial	0.529	0.497	0.349	n/a	207,526
PCL	**0.560**	**0.521**	**0.320**	**44.47%**	**(-20.0%) 166,109**
KM	0.536	0.493	0.792	99.50%	(+30.8%) 271,531
RS	0.465	0.424	0.391	99.83%	(+76.5%) 366,365

increases, the retrieval accuracy is first increased as the noise is filtered out, then decreased as the key information is affected by pruning. At $\theta_Q = 0.02$ for the UKBench dataset, and $\theta_Q = 0.015$ for the Holidays dataset, the retrieval accuracy reaches its peak, and the query cost reduction is about 70%. At $\theta_Q = 0.04$, the retrieval accuracy is still comparable to the initial Flickr60K codebook without codebook updating and index pruning, but the query cost is reduced by more than 95%.

6.5 Discussions

To implement a scalable retrieval system in P2P networks, we have to balance between various perspectives rather than focus on retrieval accuracy only. Our experimental results show that the proposed design fulfills this goal completely. The workload balance, as indicated by the Gini coefficient, is much better than the one reported in [23] (0.18–0.33 vs. 0.40–0.57), although they use different experimental datasets. The retrieval accuracy is comparable to many of the state-of-the-art quantization methods [47] (for the Holidays dataset, 0.545–0.619 vs. 0.526–0.653). While we are not aware of any literature that investigates the updating and query cost in a similar setting, the cost of our proposed method is low enough to implement a scalable system.

Finally, while there exist various techniques to improve different perspectives of the BoVW process, such as Hamming embedding (HE) and weak geometrical consistency (WGC) [8], in general, implementing them in P2P networks is not straightforward. Moreover, unlike codeword partitioning, it is not clear how the workload partitioning and query network cost is affected by these techniques. Therefore, we focus on optimizing codeword partitioning in this paper and leave them as future work.

7 Conclusion and Future Work

In this paper we present a bag-of-visual-words (BoVW) model based approach for content based image retrieval (CBIR) in peer-to-peer (P2P) networks. In order to overcome the difficulty in generating and maintaining a global codebook when the BoVW model is deployed in P2P networks, we formulate the problem of updating an existing codebook as optimizing the retrieval accuracy and workload balance. As a result, the proposed approach is scalable to the number of images shared within a P2P network and the evolving nature of P2P networks. In order to further improve the retrieval performance of the proposed approach and reduce network cost, indexing pruning techniques are applied. We conduct comprehensive experiments to evaluate various aspects of the proposed approach while demonstrating its promising performance.

In the future, we will investigate DHT specific optimizations for cost reduction, more advanced matching refinement and multi-modal fusion techniques in P2P networks, and extensions of this approach to other distributed architectures. In particular, for the CAN network [4], we can embed the index into the CAN overlay. That is, we make the CAN address space corresponding to our feature space, and replace the CAN zones with codeword partitions. Such an embedding will eliminate the overhead of an additional DHT layer, as we can implement the SPLIT/MERGE operations as a CAN zone split/takeover, instead of adding and removing entries on DHT.

Acknowledgments

This work was partially supported by ARC (Australian Research Council) grants, the NCI (National Computational Infrastructure) National Facility at the Australian National University, and HPC (High Performance Computing) service at The University of Sydney.

References

[1] M. Steiner, T. En-Najjary, and E. W. Biersack, "Long term study of peer behavior in the KAD DHT," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1371–1384, Oct. 2009.
[2] H. Schulze and K. Mochalski, "Internet study 2008/2009," Internet Studies, ipoque, 2009.
[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001, pp. 149–160.
[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001, pp. 161–172.
[5] M. Mordacchini, L. Ricci, L. Ferrucci, M. Albano, and R. Baraglia, "Hivory: Range queries on hierarchical voronoi overlays," in *IEEE International Conference on Peer-to-Peer Computing*, Aug. 2010, pp. 1–10.
[6] Y. Tang, S. Zhou, and J. Xu, "LIGHT: A query-efficient yet low-maintenance indexing scheme over DHTs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 1, pp. 59–75, Jan. 2010.
[7] L. Zhang, Z. Wang, and D. Feng, "Efficient high-dimensional retrieval in structured P2P networks," in *IEEE International Conference on Multimedia and Expo Workshops*, Jul. 2010, pp. 1439–1444.
[8] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *International Journal of Computer Vision*, vol. 87, pp. 316–336, 2010.
[9] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *IEEE International Conference on Computer Vision*, vol. 2, 2003, pp. 1470–1477.
[10] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *ACM International Workshop on Multimedia Information Retrieval*, 2007, pp. 197–206.
[11] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *European Conference on Computer Vision*, 2002, pp. 128–142.
[12] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157.
[13] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *USENIX Annual Technical Conference*. Boston, MA, USA, 2004, pp. 127–140.
[14] D. Belson, "The state of the internet, 2nd quarter, 2013 report," Akamai, 2013.
[15] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance feedback: A power tool for interactive content-based image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 644–655, Sep. 1998.
[16] C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2003, pp. 175–186.
[17] P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," in *ACM/IFIP/USENIX International Conference on Middleware*, 2003, pp. 21–40.
[18] Q. Xu, H. T. Shen, Y. Dai, B. Cui, and X. Zhou, "Achieving effective multi-term queries for fast DHT information retrieval," in *International Conference on Web Information Systems Engineering (WISE)*, 2008, pp. 20–35.
[19] H. Chen, J. Yan, H. Jin, Y. Liu, and L. M. Ni, "TSS: Efficient term set search in large peer-to-peer textual collections," *IEEE Transactions on Computers*, vol. 59, no. 7, pp. 969–980, 2010.
[20] M. Batko, D. Novak, F. Falchi, and P. Zezula, "Scalability comparison of peer-to-peer similarity search structures," *Future Generation Computer Systems*, vol. 24, no. 8, pp. 834 – 848, 2008.
[21] M. Bawa, T. Condie, and P. Ganesan, "LSH forest: Self-tuning indexes for similarity search," in *International Conference on World Wide Web*. New York, NY, USA: ACM, 2005, pp. 651–660.
[22] D. Li, J. Cao, X. Lu, and K. C. Chan, "Efficient range query processing in peer-to-peer systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 1, pp. 78–91, Jan. 2009.
[23] P. Haghani, S. Michel, and K. Aberer, "Distributed similarity search in high dimensions using locality sensitive hashing," in *International Conference on Extending Database Technology: Advances in Database Technology*. New York, NY, USA: ACM, 2009, pp. 744–755.
[24] M. Zhou, H. T. Shen, X. Gong, W. Qian, and A. Zhou, "Personalized query evaluation in ring-based P2P networks," *Information Sciences*, vol. 220, no. 0, pp. 463–482, 2013.
[25] C. Böhm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases," *ACM Computing Surveys*, vol. 33, pp. 322–373, Sep. 2001.
[26] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 194–205.
[27] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, 2007.
[28] A. Joly and O. Buisson, "Random maximum margin hashing," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 873–880.
[29] M. R. Trad, A. Joly, and N. Boujemaa, "Distributed KNN-graph approximation via hashing," in *ACM International Conference on Multimedia Retrieval*. New York, NY, USA: ACM, 2012, pp. 43:1–43:8.
[30] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2161–2168.
[31] S. Büttcher and C. L. A. Clarke, "A document-centric approach to static index pruning in text retrieval systems," in *ACM Interna-*

tional Conference on Information and Knowledge Management. New York, NY, USA: ACM, 2006, pp. 182–189.
[32] A. Moffat, W. Webber, J. Zobel, and R. Baeza-Yates, "A pipelined architecture for distributed text query evaluation," *Information Retrieval*, vol. 10, no. 3, pp. 205–231, 2007.
[33] R. Ji, L.-Y. Duan, J. Chen, L. Xie, H. Yao, and W. Gao, "Learning to distribute vocabulary indexing for scalable visual search," *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 153–166, 2013.
[34] L. Zhang, Z. Wang, and D. Feng, "Content-based image retrieval in P2P networks with bag-of-features," in *IEEE International Conference on Multimedia and Expo Workshops*, Jul. 2012, pp. 133–138.
[35] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 1794–1801.
[36] S. Lazebnik and R. Raginsky, "Supervised learning of quantizer codebooks by information loss minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1294–1309, Jul. 2009.
[37] Q. Qiu, Z. Jiang, and R. Chellappa, "Sparse dictionary-based representation and recognition of action attributes," in *IEEE International Conference on Computer Vision*, Nov. 2011, pp. 707–714.
[38] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.
[39] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European Conference on Computer Vision*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., vol. 6314. Springer Berlin Heidelberg, 2010, pp. 143–156.
[40] S. Kullback, *Information Theory and Statistics*. Dover Publications, 1997.
[41] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for IDF," *Journal of Documentation*, vol. 60, pp. 503–520, 2004.
[42] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, Mar. 2003.
[43] A. Ntoulas and J. Cho, "Pruning policies for two-tiered inverted index with correctness guarantee," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007, pp. 191–198.
[44] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol. 65, pp. 43–72, 2005.
[45] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *European Conference on Computer Vision*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 304–317.
[46] A. K. Sen, *On Economic Inequality*. Clarendon Press, 1973.
[47] R. Arandjelović and A. Zisserman, "All about VLAD," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

Lelin Zhang (S12) received the B.Eng. degree in computer science from South China University of Technology, Guangzhou, China, in 2007, and the Master of Information Technology and Information Technology Management degrees from the University of Sydney, Sydney, Australia, in 2009 and 2010, respectively. He is currently pursuing the Ph.D. degree at the Biomedical & Multimedia Information Technology (BMIT) Research Group, School of Information Technologies, University of Sydney, Sydney, Australia. His research interests include multimedia content analysis, information retrieval, computer vision, and distributed computing.

Zhiyong Wang (M05) received the B.Eng. and M.Eng. degrees in electronic engineering from South China University of Technology, Guangzhou, China, and the Ph.D. degree from Hong Kong Polytechnic University, Hong Kong. He is currently a Senior Lecturer with the School of Information Technologies, University of Sydney, Sydney, Australia, and Associate Director of the Multimedia Laboratory, University of Sydney, Sydney, Australia. His research interests focus on multimedia computing, including multimedia information processing, retrieval and management, Internet-based multimedia data mining, human-centred multimedia computing, and pattern recognition. He has published more than 70 scholarly research papers.

Tao Mei (M06-SM11) is a Lead Researcher with Microsoft Research, Beijing, China. He received the B.E. degree in automation and the Ph.D. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively. His current research interests include multimedia information retrieval and computer vision. He has authored or co-authored over 150 papers in journals and conferences, 10 book chapters, and edited three books. He holds 11 U.S. granted patents and more than 20 in pending.

Dr. Mei was the recipient of several paper awards from prestigious multimedia conferences and journals, including the IEEE TRANS. ON MULTIMEDIA Prize Paper Award in 2013, the Best Paper Awards at ACM Multimedia in 2007 and 2009, and the Best Student Paper Award at the IEEE VCIP in 2012, etc. He is an Associate Editor of the IEEE TRANS. ON MULTIMEDIA, ACM/Springer Multimedia Systems, Neurocomputing, and a Guest Editor of five international journals. He is the General Co-chair of ICIMCS 2013, the Program Co-chair of MMM 2013 and IEEE MMSP 2015. He is a Senior Member of the IEEE and the ACM.

David Dagan Feng (F03) received the M.Eng. degree in electrical engineering and computer science (EECS) from Shanghai Jiao Tong University, Shanghai, China, in 1982, and the M.Sc. degree in biocybernetics and the Ph.D. degree in computer science from the University of California, Los Angeles (UCLA), Los Angeles, CA, USA, in 1985 and 1988, respectively, where he received the Crump Prize for Excellence in Medical Engineering. He is Head of School of Information Technologies, Director of the Biomedical & Multimedia Information Technology Research Group, and Research Director of the Institute of Biomedical Engineering and Technology at the University of Sydney, Sydney, Australia. He has published over 700 scholarly research papers, pioneered several new research directions, and made a number of landmark contributions in his field. More importantly, however, is that many of his research results have been translated into solutions to real-life problems and have made tremendous improvements to the quality of life for those concerned. He has served as Chair of the International Federation of Automatic Control (IFAC) Technical Committee on Biological and Medical Systems, has organized/chaired over 100 major international conferences/symposia/workshops, and has been invited to give over 100 keynote presentations in 23 countries and regions. Prof. Feng is a Fellow of IEEE and Australian Academy of Technological Sciences and Engineering.