Zeng M, Yao B, Wang ZJ *et al.* CATIRI: An efficient method for content-and-text based image retrieval. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 34(2): 287–304 Mar. 2019. DOI 10.1007/s11390-019-1911-2

# CATIRI: An Efficient Method for Content-and-Text Based Image Retrieval

Mengqi Zeng[1], Bin Yao[1,*], *Member, CCF, ACM, IEEE*, Zhi-Jie Wang[2,3,4], *Member, CCF, ACM*
Yanyan Shen[1], Feifei Li[5], *Senior Member, IEEE, Member, ACM*, Jianfeng Zhang[6], Hao Lin[6]
and Minyi Guo[1], *Fellow, CCF, IEEE, Member, ACM*

[1] *Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[2] *School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China*
[3] *Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou 510006, China*
[4] *National Engineering Laboratory for Big Data Analysis and Applications, Beijing 100871, China*
[5] *School of Computing, University of Utah, Salt Lake City 84112, U.S.A.*
[6] *Alibaba Group, Hangzhou 311121, China*

E-mail: zmq19950714@126.com; yaobin@cs.sjtu.edu.cn; wangzhij5@mail.sysu.edu.cn; shen-yy@cs.sjtu.edu.cn
lifeifei@cs.utah.edu; {xingdian, bixuan}@taobao.com; guo-my@cs.sjtu.edu.cn

Received July 9, 2018; revised January 24, 2019.

**Abstract** The combination of visual and textual information in image retrieval remarkably alleviates the semantic gap of traditional image retrieval methods, and thus it has attracted much attention recently. Image retrieval based on such a combination is usually called the content-and-text based image retrieval (CTBIR). Nevertheless, existing studies in CTBIR mainly make efforts on improving the retrieval quality. To the best of our knowledge, little attention has been focused on how to enhance the retrieval efficiency. Nowadays, image data is widespread and expanding rapidly in our daily life. Obviously, it is important and interesting to investigate the retrieval efficiency. To this end, this paper presents an efficient image retrieval method named CATIRI (content-and-text based image retrieval using indexing). CATIRI follows a three-phase solution framework that develops a new indexing structure called MHIM-tree. The MHIM-tree seamlessly integrates several elements including Manhattan Hashing, Inverted index, and M-tree. To use our MHIM-tree wisely in the query, we present a set of important metrics and reveal their inherent properties. Based on them, we develop a top-$k$ query algorithm for CTBIR. Experimental results based on benchmark image datasets demonstrate that CATIRI outperforms the competitors by an order of magnitude.

**Keywords** image retrieval, text-and-visual feature, indexing, top-$k$

## 1 Introduction

With the booming development of electronic information technology and the network communication technology, massive image data are generated every day, mainly from social networks (e.g., Facebook, Twitter and Flickr) and general image databases (e.g., Google Images and Baidu Images)[1,2]. The scale of image databases is now in hundreds of millions, and is expanding rapidly[1,3,4]. Image retrieval as a fundamental operation in image database plays a significant role[5,6].

One of representative branches is the text-based image retrieval (TBIR)[7,8]. It is based on annotations

Regular Paper

Special Section of NSFC Joint Research Fund for Overseas Chinese Scholars and Scholars in Hong Kong and Macao 2014–2017

This work was supported by the National Basic Research 973 Program of China under Grant No. 2015CB352403, the National Key Research and Development Program of China under Grant Nos. 2018YFC1504504, 2016YFB0700502 and 2018YFB1004400, the National Natural Science Foundation of China under Grant Nos. 61872235, 61729202, 61832017, U1636210, 61832013, 61672351, 61472453, 61702320, U1401256, U1501252, U1611264, U1711261, U1711262, U61811264, and Guangdong Province Key Laboratory of Popular High Performance Computers of Shenzhen University under Grant No. SZU-GDPHPCL2017.

*Corresponding Author

associated with images, and the relevant images are retrieved from the databases by matching the textual query with the textual annotations of images[8,9]. TBIR is simple and fast, and has been widely used on the Internet (e.g., many social image search engines). Yet, TBIR bears some limitations due to manual labelling, e.g., expensive labor and time cost, covering a limited part of semantic interpretations on the image content, inconsistent and/or biased annotations[1]. To alleviate these issues, some automatic image annotation techniques have been proposed[9].

Another branch is the content-based image retrieval (CBIR)[10,11]. It is based on low-level visual properties (e.g., color, texture, sketch) extracted from the images, and in CBIR systems the similarity of two images is measured by these visual features[10,12]. CBIR thoroughly overcomes the disadvantages of TBIR, since it does not use annotations. Hence, much attention has been attracted to CBIR. Some researches are devoted to the retrieval quality (see, e.g., [10, 13, 14]), while others are devoted to the retrieval efficiency (see, e.g., [3, 5, 6, 11, 15, 16]), since CBIR is much more time-consuming than TBIR. Nevertheless, there exists a critical issue[1]: the semantic gap between low-level visual features and high-level semantic features. Typically, the semantic gap refers to the mismatch between the information extracted from an image and the interpretation for the users[13]. Two major reasons incur semantic gap[1,13]: one is the impossibility for users to describe the query image exactly and adequately, and the other is the incomprehension about users' intention behind the query.

The researches[17,18] have shown that combining textual and visual information even with simple fusion strategies may improve the quality of retrieval results. Thereby, it has attracted a lot of attention in recent years[2,18–21], and typically it is called the content-and-text based image retrieval (CTBIR). In existing literature, efforts for CTBIR are mostly on improving the retrieval quality by exploiting various techniques such as latent semantic kernels[20], semantic combination[21], and composite correlation quantization model[2]. To the best of our knowledge, few attention has focused on the retrieval efficiency in CTBIR. One can easily understand that CTBIR is also time-consuming, since we need to consider both textual and visual information, let alone the increasing scale of image databases. In view of the facts above, we make an attempt to investigate the retrieval efficiency of CTBIR.

Specifically, this paper proposes an efficient CTBIR method dubbed as CATIRI, which can preserve the quality of retrieval results and is designed for high efficiency. The key principle of CATIRI is to model the top-$k$ image retrieval problem as a problem similar to spatial-keyword query, and solve the reduced problem by exploiting 1) a new indexing structure called MHIM-tree that can be viewed as a seamlessly blend of several existing techniques such as Manhattan hashing[22], inverted index[23] and M-tree[24], and 2) a set of important properties that allow us to prune most of nodes in the query processing. As we demonstrate with benchmark datasets, CATIRI is significantly superior to the baseline. To summarize, the novelty and contributions of this paper are as follows.

• We reveal that the retrieval efficiency for CBTIR is also urgently important yet it is widely ignored. This paper makes the first deep investigation on the retrieval efficiency of CBTIR.

• We show the top-$k$ image retrieval problem can be modelled as a problem similar to the spatial-keyword query. This finding could open a hopeful direction for image retrieval.

• We present a top-$k$ image retrieval method, CATIRI, that yields a new indexing structure and reveals a set of important properties.

• We give rigorous theoretical analysis for our method, and conduct comprehensive experiments to evaluate its performance.

The rest of the paper is organized as follows. Section 2 reviews previous work most related to ours. Section 3 formulates our problem, and Section 4 presents our indexing structure MHIM-tree. In Section 5 several important metrics and their inherent properties are examined, and Section 6 presents our three-phase solution. Section 7 presents the theoretical analysis for our proposed indexing as well as the retrieval algorithm. In Section 8 we experimentally evaluate our proposed solution, and finally Section 9 concludes the paper.

## 2 Related Work

Image retrieval is a classic and hot topic in tens of years. We review existing image retrieval methods most related to ours. Generally, they can be classified into three categories: 1) text-based image retrieval (TBIR), 2) content-based image retrieval (CBIR), and 3) content-and-text based image retrieval (CTBIR).

*TBIR.* In TBIR, images are annotated with text that represents high-level semantics, and image retrieval is performed by text retrieval techniques. For example, Zhang *et al.*[7] suggested a user-term feedback based technique for text-based image retrieval. Li *et al.*[8] proposed an approach to learn a robust classifier for text-based image retrieval. TBIR is simple and fast yet manual labelling is a labor intensive process[25], and so many researches focus on automatic image annotation techniques[9].

*CBIR.* Correspondingly, in CBIR, image content is used to measure similarity, which is described by visual features such as color, shape and texture. CBIR overcomes the disadvantages of manual labelling, but it is not so satisfactory[13] due to the semantic gap between low-level visual features and high-level semantics. Many researches have been devoted to narrowing or bridging the semantic gap, and various techniques and algorithms are proposed. For example, Tong and Chang[10] proposed a support vector machine active learning algorithm for content-based image retrieval. Yang and Lozano-Perez[26] suggested multiple-instance learning techniques for image retrieval. Deng *et al.*[14] incorporated prior human knowledge in the form of a hierarchical structure to perform content-based image retrieval.

Compared with TBIR, CBIR is much more time-consuming. A lot of studies have been devoted to the efficiency in CBIR. For example, Xia *et al.*[16] developed multi-kernel locality sensitive hashing schema, significantly improving the retrieval performance of KLSH[12] by utilizing multiple kernels. Natsev *et al.*[27] proposed WALRUS retrieval algorithm, which extracts features of region and computes their signature as an index to measure similarity. Shen *et al.*[5] addressed how to speed up interactive image retrieval, where a query is interactively refined towards the optimal answers by exploiting user feedback. Falchi *et al.*[6] studied the possibility of caching the answers to content-based image retrieval queries in metric space, with the aim of reducing the average cost of query.

Although these studies are related to our work, it can be seen that they are clearly different from ours, since this paper mainly focuses on the content-and-text based image retrieval, instead of CBIR.

*CTBIR.* In CTBIR, it can take advantage of both visual and textual information, and bridge the semantic gap. Thereby, it has attracted a lot of attention in recent years[2,18–21]. For example, Zagoris *et al.*[19] proposed a two-stage approach, which first uses a text modality to rank the collection and then performs CBIR only on the top-$k$ items. Long *et al.*[2] proposed a composite correlation quantization model for image retrieval. Their model jointly finds correlation-maximal mappings, and learns composite quantizers that convert the isomorphic latent features into compact binary codes. Zhou and Huang[18] proposed a seamless joint querying and relevance feedback scheme, based on both keywords and low-level visual contents. Caicedo *et al.*[20] proposed a strategy to fuse visual features and unstructured-text data in a medical image retrieval system. Clinchant *et al.*[21] proposed a set of techniques called semantic combination that better manages the complementariness between text and image search systems.

These studies are mainly devoted to improving the retrieval quality by developing various fusion techniques. To the best of our knowledge, less attention has been focused on the efficiency in CTBIR. This paper makes an attempt to address the efficiency issue in CTBIR. Remark that the demo paper[28] may be the closest to our work, since it mentions the retrieval efficiency by directly exploiting an existing technique "inverted index". Nevertheless, its focus is still on the retrieval quality, as indicated in their experiments. Essentially, we use a similar version of method in [28] as the baseline. Our solution CATIRI achieves excellent query performance, and is significantly superior to the baseline.

*Others.* There are also other studies that are close to but clearly different from our work. For example, Rabitti and Savino[29] presented an approach to image retrieval that allows to take into account the imprecision assigned to the different parts of the query, and to rank the images on the basis of this imprecision. Chu *et al.*[30] introduced a semantic data model to capture the hierarchical, spatial, temporal, and evolutionary semantics of images in pictorial databases. Brown and Gruenwald[31] proposed a prototype content-based image retrieve system that aims to save space cost. Chen *et al.*[32] developed a distributed retrieval and recommendation system for geo-textual images.

## 3 Preliminaries

### 3.1 Problem Formulation

Let $D$ be an image database, and each image object $I$ in $D$ is defined as a pair $(I_v, I_t)$, where $I_v$ is a vector of visual features (e.g., GIST[33] and bag-of-words[34]), and $I_t$ is a document consisting of textual annotations. Usually, document $I_t$ can be represented by a vector, in which each dimension is the weight of a distinct term in the document[35,36].

Let $D_c$ be the collection of all documents in database $D$. The weight of a term $t$ in $I_t$ can be computed as follows[36]:

$$w(I_t, t) = p(t|I_t) = (1-\lambda)\frac{tf(t, I_t)}{|I_t|} + \lambda\frac{tf(t, D_c)}{|D_c|}, \quad (1)$$

where $tf(t, I_t)$ (resp., $tf(t, D_c)$) is the term frequency of term $t$ in $I_t$ (resp., $D_c$), $|I_t|$ (resp., $|D_c|$) is the the total number of tokens in document $I_t$ (resp., all documents of $D_c$), and $\lambda \in [0, 1]$ is a smoothing coefficient of the Jelinek-Mercer method[36]. Notice that here $tf(t, I_t)/|I_t|$ (resp., $tf(t, D_c)/|D_c|$) is essentially the maximum likelihood estimate of term $t$ in $I_t$ (resp., $D_c$).

Let $Q$ be a query defined as a pair $(Q_v, Q_k)$, where $Q_v$ is a vector of visual features, and $Q_k$ is a set of query keywords. The text relevancy, between a given query $Q$ and an image object $I$, can be regarded as the probability that query keywords $Q_k$ appear in the document $I_t$. And it can be computed as follows:

$$P(Q_k|I_t) = \prod_{t \in Q_k} p(t|I_t). \quad (2)$$

Correspondingly, the visual similarity between a given query $Q$ and an image object $I$ can be usually obtained by computing the distance between their visual vectors. That is,

$$vDist(Q_v, I_v) = \|Q_v - I_v\|.$$

Furthermore, the similarity score between an image object $I$ and the query $Q$ can be computed as follows by linear fusion technique[37]:

$$S(Q, I) = \alpha N(S_v(Q, I)) + (1-\alpha)N(S_t(Q, I)), \quad (3)$$

where $S_v(\cdot)$ represents the visual similarity between $Q$ and $I$, $S_t(\cdot)$ represents the text relevancy between $Q$ and $I$, $N(\cdot)$ is a normalization operator that transforms them into a range $[0, 1]$, and $\alpha \in [0, 1]$ is a balance parameter between the visual similarity and the text relevancy. Note that, the similarity score described above is a linear combination of text relevance and visual similarity. Nevertheless, our solution can also work for other functions, since it is independent of the fusion functions. Formally, in this paper we focus on the following problem.

**Definition 1** (Top-$k$ Image Retrieval). *Given an image database $D$ and a query $Q$, the top-$k$ image retrieval is asked to return $k$ image objects that are ranked the highest in terms of the similarity scores.*

## 3.2 Problem Analysis

It is widely accepted[1,12,16] that feature hashing can map visual features of images into the compact binary codes. Most of existing hashing methods adopt Hamming distance to measure the similarity between points in the hashcode space. Yet, these hashing methods may destroy the neighborhood structure in the original feature space, violating the essential goal of feature hashing[38].

Recently, a new feature hashing technique called Manhattan Hashing[22] shows that it can effectively preserve the neighborhood structure in the original feature space. Nevertheless, their work mainly focuses on the retrieval quality for the content-based image retrieval (CBIR). Note that in this paper we are interested in the content-and-text based image retrieval (CTBIR), and our goal is to achieve a remarkable improvement on the retrieval efficiency.

Nevertheless, Manhattan hashing is still a powerful tool for our work. It enlightens us to re-examine our problem, and particularly we realize that, by using Manhattan hashing, it may allow us to transform our retrieval problem into a new version that still preserves the inherent equivalence. The followings show the reduction.

The Manhattan distance between $Q_v$ and $I_v$ in the hashcode space is denoted by $Dist(Q_v, I_v)$. One can rewrite (3) as follows:

$$S(Q, I) = \alpha(1 - \frac{Dist(Q_v, I_v)}{maxD}) + (1-\alpha)\frac{P(Q_k|I_t)}{maxP}, \quad (4)$$

where $maxD$ is the upper bound on $Dist(Q_v, I_v)$ and is used to normalize $Dist(Q_v, I_v)$ into $[0, 1]$, while $maxP$ is the upper bound on $P(Q_k|I_t)$ and is used to normalize $P(Q_k|I_t)$ into $[0, 1]$.

This reduction is important, since it is the cornerstone of our proposed method. In particular, we observe that to some extent the reduced problem is similar to spatial-keyword query problems[39], since they also involve text and have two components in the query input. This mightily motivates us to develop solutions by integrating Manhattan hashing[22] with the techniques widely used in keyword query such as inverted index[23], and used in similarity search such as M-tree[24]. To the best of our knowledge, to date 1) none of existing spatial-keyword query methods can solve our problem, and 2) this is the first work to expose that the top-$k$ image retrieval problem can be modelled as a problem similar to the spatial-keyword query. In what follows, we present our solution including indexing structure, important metrics, and specific algorithms. For ease of reference, Table 1 gives main symbols in this paper.

**Table 1.** Notations

| Notation | Meaning |
|---|---|
| $D/D_c$ | Image database/collection of documents in $D$ |
| $I/I_v/I_t$ | Image object/visual features vector of $I$/document of $I$ |
| $w(I_t, t)$ | Weight of term $t$ in document $I_t$ |
| $Q/Q_v/Q_k$ | Top-$k$ query/visual features vector of $Q$/keywords of $Q$ |
| $P(Q_k\|I_t)$ | Text relevancy between $Q_k$ and $I_t$ |
| $vDist(Q_v, I_v)$ | Distance between $Q_v$ and $I_v$ |
| $Dist(Q_v, I_v)$ | Manhattan distance between the hashcodes of $Q_v$ and $I_v$ |
| $S_v(\cdot)/S_t(\cdot)$ | Visual similarity/text relevancy |
| $S(\cdot)/sDif(\cdot)$ | Similarity score/change to insert an entry |
| $R/rDif(\cdot)$ | Minimum bounding sphere of entry/enlargement of the covering radius of $R$ |
| $\mathbb{V}/vDif(\cdot)$ | Document vector of image object or node/difference between vectors |
| $E/C$ | Entry in a node/category label(s) |

## 4 MHIM-Tree

In this section, we present a novel indexing structure, named MHIM-tree. Similar to many existing indexing structures such as sequenced multi-attribute tree[15], our MHIM-tree is also a blend of several existing techniques (with some specific adaptations). In a nutshell, our MHIM-tree integrates several elements including Manhattan hashing[22], inverted index[23], and M-tree[24]. For ease of presentation, we next shortly review these three existing elements. And then we examine the details of the MHIM-tree.

### 4.1 Three Existing Elements

*Manhattan Hashing.* As mentioned earlier, it is mainly used to transform the visual features of images into binary Manhattan hashcodes that preserve the neighborhood structure in the original feature space. More specifically, it first uses the iterative quantization (ITQ)[40] to project the original data with higher dimension to the lower dimensional space, and then uses Manhattan quantization (MQ)[22] to get the corresponding hashcode, such as 110011 (a 3-dimensional hashcode).

*Inverted Index.* It has a vocabulary of all terms, and each term is associated with an inverted list. Each inverted list comprises a sequence of postings, each of which normally contains the identifier of an object whose description contains the term and the frequency of the term in the description. In general, the postings in each inverted list are sorted by the object's identifier.

*M-Tree.* Similar to B-tree, M-tree is also a balanced search tree, mainly used for indexing generic multi-dimensional "metric space", where the distance function satisfies the symmetry, non-negativity and triangle inequality postulates. The key idea is to group nearby objects and represent them with their minimum bounding sphere in the next higher level of the tree. Since all objects lie within this bounding sphere, a query that does not intersect the bounding sphere also cannot intersect any of the contained objects. At the leaf level, each indexed object is represented by a sphere, where the sphere center is the feature of the object and the radius is zero. At higher levels, the aggregation of an increasing number of objects is called routing object, represented by a bounding sphere of contained objects.

### 4.2 Data Structure of MHIM-Tree

Based on M-tree, MHIM-tree is a high-balanced tree with the branching factor $B$. For a leaf node, it contains entries in the form of $(id, R, \mathbb{V}, C)$, and each entry represents an image object.

• $id$: the identifer of an image object $I$.

• $R(O, r)$: the minimum bounding sphere of image object $I$, whose center $O$ is the hashcode of $I$ and the radius is zero. Remark that the sphere for a single image object is essentially a point. For example, assume the 3-dimensional hashcode of $I$ is 001101, then $R.O = (0, 3, 1)$ and $R.r = 0$.

• $\mathbb{V}$: a vector that represents the document $I_t$. For example, assume that there are $n$ distinct terms $t_1, t_2, ..., t_n$ in $D_c$, then $\mathbb{V} = (v_1, v_2, ..., v_n)$, where $v_i = w(I_t, t_i)$ (i.e., $p(t_i|I_t)$, recall (1)) if $t_i \in I_t$; otherwise $v_i = 0$.

• $C$: a label for the category containing $I$. Note that, in general all images in the database $D$ are divided into many categories.

For a non-leaf node, it contains entries in the form of $(p, R, \mathbb{V}, C)$, and each entry represents a routing object corresponding to a child node. Note that, here we slightly abuse the notations $R, \mathbb{V}, C$, but their meanings shall be clear in the context.

• $p$: a pointer to the corresponding child node.

• $R(O, r)$: the minimum bounding sphere of the routing object that bounds the spheres of all entries in the child node.

• $\mathbb{V}$: the representative vector of all the document vectors in the subtree rooted at the child node. For example, assume $\mathbb{V} = (v_1, v_2, \cdots, v_n)$, then $v_j = \max\{v_j \text{ of } E_i.\mathbb{V} | E_i \in p\}$, where $E_i$ is an entry contained in the child node that $p$ points to.

• $C$: a set of all labels for categories in the child node. That is, $C = \bigcup\{E_i.C\}$.

In addition, each node $\mathcal{N}$ (regardless of leaf or non-leaf node) is attached with an inverted index, which is used to index all the documents in the subtree rooted at $\mathcal{N}$. The inverted index is composed of two main components:

• a vocabulary of all distinct terms in documents that are in the subtree rooted at $\mathcal{N}$;

• an inverted file storing inverted lists, each of which is linked to a term $t$ in the vocabulary. Note that, here each inverted list is a set of tuples in the form of $\langle E, c, w((E, c), t)\rangle$, where $E$ refers to an entry in the node containing term $t$, $c$ is one of category labels in entry $E$, and $w((E, c), t)$ is the weight of term $t$ in both entry $E$ and category $c$.

Remark that, when $\mathcal{N}$ is a leaf node, $c = E.C$ and $w((E, c), t) = w(E.id, t)$, since entry $E$ represents an image object. Otherwise, $c$ is a category label in the set $E.C$, and $w((E, c), t)$ is the maximum weight of term $t$ in all documents of image objects that belong to the subtree rooted at $E$ and category $c$, since entry $E$ represents a child node of $\mathcal{N}$. Here $w((E, c), t)$ is computed as follows:

$$w((E, c), t) = \max\{w((E_i, c), t) | E_i \in E.p\}, \quad (5)$$

where $E_i$ is an entry contained in the node represented by $E$.

*Example* 1. To further understand the general picture of the MHIM-tree, Fig.1 shows a general picture of our MHIM-tree. Here $I_1, I_2, \ldots, I_9$ are nine image objects, which are grouped into four different categories $C_1$, $C_2$, $C_3$, and $C_4$. And *inv*1–*inv*8 are the inverted index.

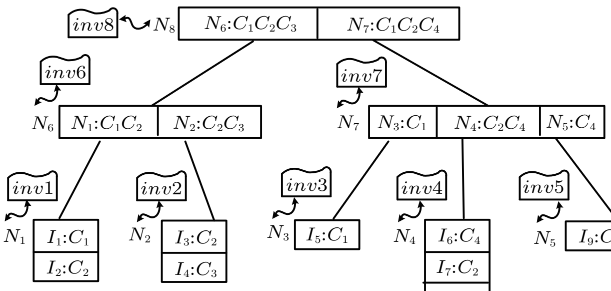


Fig.1. Index structure of MHIM-tree.

## 5 Important Metrics and Properties

In order to use MHIM-tree wisely in the retrieval, this section defines a set of important metrics, and meanwhile examines their inherent properties.

As we know, for two points $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$, $\boldsymbol{y} = (y_1, y_2, \cdots, y_d)$ in the Manhattan space, the Manhattan distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is the sum of differences on all dimensions. The specific calculation formula is as follows[22]:

$$Dist(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} |x_i - y_i|. \quad (6)$$

Obviously, the Manhattan distance satisfies the symmetry, non-negativity, and triangle inequality postulates.

Consider, in the Manhattan hashcode space, a point $P = (p_1, p_2, \ldots, p_d)$ and a sphere $R = (O, r)$, where $O = (o_1, o_2, \ldots, o_d)$ is the center of $R$ and $r$ is the radius. The first metric is as follows.

**Definition 2**. *Given $P$ and $R = (O, r)$, the minimum Manhattan distance between $P$ and $R$ in Manhattan hashcode space, is defined as follows*:

$$minDist(P, R) = \max(Dist(P, O) - r, 0).$$

The metric above has the following property.

**Lemma 1**. *The distance $minDist(P, R)$ is no larger than the distance between $P$ and any object enclosed in $R$.*

*Proof.* Let $S$ be any image object enclosed in $R$, the hashcode of which is $(s_1, s_2, \ldots, s_d)$. It is obvious that $\forall i \in [1, d]$, we have

$$|p_i - s_i| \geqslant |p_i - o_i| - |s_i - o_i|.$$

By (6), we can obtain

$$Dist(P, S) \geqslant Dist(P, O) - Dist(S, O).$$

And $Dist(S, O) \leqslant r$, therefore

$$Dist(P, S) \geqslant Dist(P, O) - r.$$

Then, by Definition 2 and non-negativity of Manhattan distance, we have

$$minDist(P, R) \leqslant Dist(P, S). \quad \square$$

The above lemma offers a lower bound on the Manhattan distance between a query and all image objects enclosed in the sphere of a node, and it shall be used in our proofs later. The second metric we present is the maximum text relevancy between a query and a node, which takes the category into consideration.

**Definition 3**. *Given a query $Q$ and a node $\mathcal{N}$, the maximum text relevancy between $Q$ and $\mathcal{N}$ is defined as follows:*

$$maxRel(Q_k,\mathcal{N}) = \max_{c\in\mathcal{N}.C} P(Q_k|(\mathcal{N},c)). \tag{7}$$

*$P(Q_k|(\mathcal{N},c))$ is an upper bound on the text relevancy between $Q_k$ and documents of all image objects that are contained in the subtree rooted at $\mathcal{N}$ and belong to category $c$. It is computed as:*

$$P(Q_k|(\mathcal{N},c)) = \prod_{t\in Q_k} p(t|(\mathcal{N},c)) = \prod_{t\in Q_k} w((\mathcal{N},c),t),$$

*where for $t\in(\mathcal{N},c)$, $w((\mathcal{N},c),t)$ is stored in the inverted index corresponding to the parent node of $\mathcal{N}$, and for $t\notin(\mathcal{N},c)$, $w((\mathcal{N},c),t)=\lambda\frac{tf(t,D_c)}{|D_c|}$.*

**Lemma 2**. *The text relevancy $maxRel(Q_k,\mathcal{N})$ is no less than the text relevancy between query $Q$ and any image object contained in the subtree rooted at node $\mathcal{N}$.*

*Proof.* Let $I$ be any image object belonging to the subtree rooted at $\mathcal{N}$, and the text relevancy between $q$ and $I$ is:

$$P(Q_k|I_t) = \prod_{t\in Q_k} w(I_t,t).$$

According to (5), we have

$$w((\mathcal{N},I.C),t) \geqslant w(I_t,t).$$

By combining the above two results, we can obtain

$$P(Q_k|(\mathcal{N},I.C)) \geqslant P(Q_k|I_t).$$

Further, according to (7) and the fact $I.C\in\mathcal{N}.C$, we have

$$maxRel(Q_k,\mathcal{N}) \geqslant P(Q_k|I_t). \qquad \square$$

The lemma above offers an upper bound on the text relevancy between a query and all image objects contained in the subtree rooted at a node, and it shall be used in the proof of Lemma 3. Furthermore, based on (4) we propose the third metric, the maximum similarity score, which combines $minDist$ and $maxRel$. Specifically, we have the following definition.

**Definition 4**. *Given a query $Q$ and a node $\mathcal{N}$, the maximum similarity score between $Q$ and $\mathcal{N}$ is defined as follows:*

$$MaxScore(Q,\mathcal{N}) = \alpha\left(1-\frac{minDist(Q_v,\mathcal{N}.R)}{maxD}\right) + (1-\alpha)\frac{maxRel(Q_k,\mathcal{N})}{maxP}, \tag{8}$$

*where $\alpha$, $maxD$ and $maxP$ are the same with those in (4).*

**Lemma 3**. *The similarity score $MaxScore(Q,\mathcal{N})$ is no less than the similarity score between any image object and query $Q$, which is contained in the subtree rooted at node $\mathcal{N}$.*

*Proof.* Let $I$ be an image object belonging to the subtree rooted at node $\mathcal{N}$, and the similarity score of $I$ for query $Q$ is $S(Q,I)$ computed by (4). By Lemma 1, we have

$$minDist(Q_v,\mathcal{N}.R) \leqslant Dist(Q_v,I_v).$$

Further, by Lemma 2, we can obtain

$$maxRel(Q_k,\mathcal{N}) \geqslant P(Q_k|I_t).$$

By (4) and (8), we have

$$MaxScore(Q,\mathcal{N}) \geqslant S(Q,I). \qquad \square$$

For a query, the lemma above offers an upper bound on the similarity scores of all image objects contained in the subtree rooted at node $\mathcal{N}$. The last metric is the maximum Manhattan distance between a point and a sphere.

**Definition 5**. *Given a point $P$ and a sphere $R=(O,r)$, the maximum Manhattan distance between $P$ and $R$ in Manhattan hashcode space is defined as:*

$$maxDist(P,R) = Dist(P,O) + r.$$

**Lemma 4**. *The distance $maxDist(P,R)$ is no smaller than the distance between $P$ and any object enclosed in $R$.*

*Proof.* Let $S$ be any object enclosed in $R$, the hashcode of which is $(s_1,s_2,\ldots,s_d)$, and it is obvious that for $\forall i\in[1,d]$, we have

$$|p_i-s_i| \leqslant |p_i-o_i| + |s_i-o_i|.$$

By (6), we can obtain

$$Dist(P,S) \leqslant Dist(P,O) + Dist(S,O).$$

And $Dist(S,O)\leqslant r$, therefore

$$Dist(P,S) \leqslant Dist(P,O) + r.$$

By Definition 5, we have

$$maxDist(P,R) \geqslant Dist(P,S). \qquad \square$$

Besides, one can easily obtain the extra results below. Given a query $Q$ and the root node, *root*, of MHIM-tree, one can see that, for any image object $I$ in the tree, it is enclosed in $root.R$. Thus, by Lemma 4 and Lemma 2, one can obtain the following corollaries.

**Corollary 1**. $maxDist(Q_v,root.R) \geqslant Dist(Q_v,I_v)$.

**Corollary 2**. $maxRel(Q_k,root) \geqslant P(Q_k|I_t)$.

The above two facts imply that, in the MHIM-tree we can essentially view $maxDist(Q_v,root.R)$ as $maxD$, and $maxRel(Q_k,root)$ as $maxP$.

## 6 CATIRI Retrieval Algorithm

CATIRI comprises several phases: 1) text-and-visual feature preprocessing; 2) building MHIM-tree; 3) image retrieval via MHIM-tree.

### 6.1 Preprocessing

In this phase, we preprocess 1) the documents of textual annotations, and 2) the visual features of image objects.

The text preprocessing is mainly for calculating the text relevancy. Specifically, for each document $I_t$, one can calculate, according to (1), the weight $w(I_t, t)$ of each distinct term $t$. In this way, one can obtain a set of term-weight pairs $(t, w(I_t, t))$. In addition, one may need to calculate $\lambda\frac{tf(t,D_c)}{|D_c|}$ for each distinct term $t$ in the database $D$. This value represents the weight of $t$ when term $t$ is not in document $I_t$. After text processing, one can easily get the text relevancy according to (2).

The visual feature preprocessing is for transforming the visual features of image objects into hashcodes. As mentioned earlier, we use Manhattan hashing algorithm[22] to get the hashcodes of visual features.

### 6.2 Building MHIM-Tree

As mentioned in Section 4, our MHIM-tree is based on the classic M-tree. This allows us to adapt the construction algorithm of M-tree to achieve our goal. Generally, the construction of MHIM-tree is similar to that of M-tree. To save space, we only present the parts that are different from M-tree.

The MHIM-tree is built by *Insert* operation that is described in Algorithm 1. In this algorithm, *ChooseLeaf* is responsible for choosing the best leaf node to place a new image object. This operation starts from the root, and recursively selects the best subtree to enclose the image object, until a leaf node is reached.

**Algorithm 1 .** *Insert*$(I, category)$

```
Input: image object $I$ and its category label $category$
1: $R \leftarrow createSphere(I_v, 0)$
2: $E \leftarrow createEntry(I.id, R, I_t, category)$
3: $\mathcal{N} \leftarrow ChooseLeaf(E)$
4: Add $E$ to node $\mathcal{N}$, add $I_t$ and $category$ to the inverted file
   corresponding to $\mathcal{N}$
5: $AdjustTree(\mathcal{N})$
```

Algorithm 2 covers more details about the *ChooseLeaf* operation. Note that the selection criterion is to make the change for inserted subtree as small as possible. Here the change includes the enlargement of the covering radius, and also the difference of the document vector. The followings show how the change is measured.

**Algorithm 2 .** *ChooseLeaf*$(E)$

```
Input: an entry $E$ representing an image object
Output: the leaf node that $E$ should be inserted into
1: $\mathcal{N} \leftarrow root$
2: while $\mathcal{N}$ is a non-leaf node do
3:     for each entry $E_i$ in $\mathcal{N}$ do
4:         $change \leftarrow sDif(E_i, E)$
5:     end for
6:     Select the entry $E_i$ with the smallest $change$
7:     $\mathcal{N} \leftarrow$ the node pointed by $E_i.p$
8: end while
9: return $\mathcal{N}$
```

Let $E$ be the entry of the image object to be inserted, and $E_i$ be one of entries in the current node. Without loss of generality, assume that one wants to insert $E$ into the subtree rooted at $E_i$, then the enlargement of the covering radius is:

$$rDif(E_i, E) = CMB(E_i.R, E.R).r - E_i.r,$$

where $CMB(E_i.R, E.R)$ is minimum bounding sphere enclosing $E_i.R$ and $E.R$, and thus its radius is:

$$\begin{aligned}&CMB(E_i.R, E.R).r\\ &= \max(Dist(E_i.O, E.O) + E.r, E_i.r).\end{aligned}$$

Correspondingly, the change of the document vector is:

$$vDif(E_i, E) = 1 - cosSim(E_i.\mathbb{V}, E.\mathbb{V}),$$

where *cosSim* is the cosine similarity between vectors.

With the above two concepts in mind, the "overall" change is measured by the following:

$$sDif(E_i, E) = \beta\frac{rDif(E_i, E)}{maxR} + (1-\beta)vDif(E_i, E), \quad (9)$$

where $maxR$ is the maximum $rDif$, which is used to normalize $rDif$ into $[0, 1]$. In addition, $\beta \in [0, 1]$ is a parameter used to balance visual similarity and text relevancy. In general, one can set $\beta = \alpha$ in order to maintain the consistency.

Another important operation in Algorithm 1 is the *AdjustTree* operation, which is responsible for adjusting the MHIM-tree after inserting a new image object. This operation starts from the leaf node, recursively adjusts the entry of each visited node and updates the inverted index in its parent node, until the root is reached. Algorithm 3 covers more details about this operation.

A key operation in Algorithm 3 is *SplitNode*, used to split the node into two new nodes when the number of entries in a node exceeds the maximum limit. The division criterion is to make more similar entries in the same node, and less similar entries in different nodes. By incorporating the above criterion (based on $sDif$ in (9)) and the Split policy in [24], Algorithm 4 covers the details about *SplitNode*.

**Algorithm 3 .** $AdjustTree(\mathcal{N})$

```
Input: a node $\mathcal{N}$
 1: $\mathcal{N}_p \leftarrow$ parent node of $\mathcal{N}$
 2: if $\mathcal{N}$ needs to be split then
 3:     $\{\mathcal{N}_1, \mathcal{N}_2\} \leftarrow SplitNode(\mathcal{N})$
 4:     if $\mathcal{N}$ is root then
 5:         Create a new node $\mathcal{N}'$
 6:         Add $\mathcal{N}_1, \mathcal{N}_2$ into $\mathcal{N}'$ as child nodes, and update the
        inverted index corresponding to $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{N}'$
 7:         root $\leftarrow \mathcal{N}'$
 8:     else
 9:         Delete $\mathcal{N}$ from the parent node $\mathcal{N}_p$, and add $\mathcal{N}_1, \mathcal{N}_2$
        to $\mathcal{N}_p$ as child nodes
10:     end if
11: else
12:     Adjust the entry of $\mathcal{N}$
13:     Update the inverted file corresponding to $\mathcal{N}_p$
14: end if
15: if $\mathcal{N}$ is not root then
16:     $AdjustTree(\mathcal{N}_p)$
17: end if
```

**Algorithm 4 .** $SplitNode(\mathcal{N})$

```
Input: a node $\mathcal{N}$ that needs to be split
Output: two new nodes obtained by splitting $\mathcal{N}$
 1: $m \leftarrow$ the minimum number of entries in a node
 2: $\mathcal{N}_1, \mathcal{N}_2 \leftarrow$ empty node
 3: Insert $E_1$ in node $\mathcal{N}$ into $\mathcal{N}_1$
 4: for each remaining entry $E_i$ in node $\mathcal{N}$ do
 5:     $sumD \leftarrow sDif(E_1, E_i)$
 6: end for
 7: Select the entry with the largest $sumD$, and insert it into $\mathcal{N}_2$
 8: while there are unassigned entries in node $\mathcal{N}$ do
 9:     if one node has to contain all the rest to reach $m$ then
10:         Insert all the rest of entries into it
11:         break
12:     end if
13:     for each remaining entry $E_i$ do
14:         $d_1 \leftarrow sDif(\mathcal{N}_1, E_i)$
15:         $d_2 \leftarrow sDif(\mathcal{N}_2, E_i)$
16:     end for
17:     Select the entry with the largest $|d_1 - d_2|$ and insert it
    into the node with smaller $sDif()$
18: end while
19: return $\{\mathcal{N}_1, \mathcal{N}_2\}$
```

## 6.3 Retrieval via MHIM-Tree

Our query algorithm follows the best-first traversal paradigm[41]. It fully exploits the proposed indexing structure as well as the metric properties. Generally speaking, we employ a priority for nodes and image objects to determine the order in which they are to be visited. For an image object, its priority is the similarity score computed by (4), and for a node its priority is the $MaxScore$ computed by (8). In the algorithm, we use the common rule: "the larger the score is, the higher the priority is". In the retrieval process, our algorithm repeatedly chooses the next node or image object from the priority queue, which keeps track of unvisited nodes and image objects, until $k$ image objects have been obtained. Algorithm 5 illustrates the details for top-$k$ image retrieval.

**Algorithm 5 .** $Query(Q, root, k)$

```
Input: a query $Q$, the root node $root$, and the number of desired
    images $k$
Output: a list of desired images
 1: $que \leftarrow createPriorityQueue()$
 2: $result \leftarrow createList()$
 3: $count \leftarrow 0$
 4: $que$.Enqueue($root$, 1)
 5: while $que$ is not empty do
 6:     $elem \leftarrow que$.Dequeue()
 7:     if $elem$ is an image object then
 8:         $result$.add($elem$)
 9:         $count{+}{+}$
10:         if $count = k$ then
11:             break
12:         end if
13:     else if $elem$ is a leaf node then
14:         for each entry (image object $I$) in $elem$ do
15:             $priority \leftarrow S(Q, I)$
16:             $que$.Enqueue($I, priority$)
17:         end for
18:     else
19:         for each entry (node $\mathcal{N}$) in $elem$ do
20:             $priority \leftarrow MaxScore(Q, \mathcal{N})$
21:             $que$.Enqueue($\mathcal{N}, priority$)
22:         end for
23:     end if
24: end while
25: return $result$
```

## 7 Theoretical Analysis

In this section, we cover main theoretical results related to our method. In analysis, we use $m/M$ to denote the minimum/maximum number of entries in a node, and $d$ to denote the dimension of hashcode.

**Theorem 1**. *Our top-k query processing algorithm is correct.*

*Proof.* To prove our query algorithm is correct, the key of point is to show that all image objects are dequeued in the descending order of their similarity scores. The followings validate this fact. Given a query $Q$, two image objects $I_1$ and $I_2$, without loss of generality, assume $I_1$ is dequeued before $I_2$ (in the query processing). It is easily to know that, after $I_1$ is dequeued, there are two cases:

1) $I_2$ is in the queue;

2) a node $\mathcal{N}$ containing $I_2$ is in the queue.

If it is the former case, then it is not hard to see that $S(Q, I_1) \geqslant S(Q, I_2)$, since $I_1$ has a higher priority than $I_2$. In contrast, if it is the latter case, we can get $S(Q, I_1) \geqslant MaxScore(Q, \mathcal{N}) \geqslant S(Q, I_2)$, according to Lemma 3. Therefore, image objects are dequeued in the descending order of similarity scores in Algorithm 5. □

**Theorem 2**. *The upper bound on the time cost of constructing an MHIM-tree containing N image objects is*

$$O(MN\lceil\log_m N\rceil(d + |D_c|)).$$

*Proof.* The height of an MHIM-tree containing $N$ image objects is at most $\lceil\log_m N\rceil - 1$, since the branch number of each node is between $m$ and $M$. And the maximum number of nodes in the tree can be estimated as

$$maxNode = \lceil\frac{N}{m}\rceil + \lceil\frac{N}{m^2}\rceil + \cdots + 1 \approx \frac{N-1}{m-1}.$$

To insert an image object, one can follow a path from the root node to the leaf node for the *Choose-Leaf* operation. For each visited node in the *Choose-Leaf* operation, we scan no more than $M$ entries to find the most appropriate one, and the cost per entry is $O(d + |D_c|)$. Clearly, the cost of *ChooseLeaf* operation should be $O((\lceil\log_m N\rceil - 1)M(d + |D_c|))$.

In addition, we need to adjust the tree by the *AdjustTree* operation, since the new node may incur some updates for the upper level nodes. The updates can be done by following the reversed path from the chosen leaf to the root. For each visited node in the *Adjust-Tree* operation, we need to adjust the attributes and the inverted index in its parent node. Hence, the cost for *AdjustTree* operation is $O((\lceil\log_m N\rceil - 1)(d + |D_c|))$.

Based on the above facts, we can know that the cost for inserting an image object (without *SplitNode* operations) is $O((\lceil\log_m N\rceil - 1)(M + 1)(d + |D_c|))$.

On the other hand, it is not hard to see that the cost for each *SplitNode* operation is $O(\frac{M^2+M}{2}(d + |D_c|))$, and the number of *SplitNode* operations is no more than $maxNode - 1$.

Thus, to sum up, the time cost of building an MHIM-tree is at most

$$O((M + 1)N(\lceil\log_m N\rceil + \frac{M}{2(m-1)})(d + |D_c|))$$
$$\approx O(MN\lceil\log_m N\rceil(d + |D_c|)).$$ □

**Theorem 3**. *For an MHIM-tree containing N image objects with c* ($\geqslant 1$) *categories, the upper bound on space cost is* $O(Nd + (\frac{c}{m} + 2)N|D_c|)$.

*Proof.* Without loss of generality, assume that the number of all nodes is *nodeSum*. It is easily verified that in the MHIM-tree there are $N$ entries that represent image objects, and *nodeSum* entries that represent nodes. The space cost per entry is $O(d + |D_c|)$, in which $O(d)$ is for $R$ in the entry and $O(|D_c|)$ is for $\mathbb{V}$ in the entry. Therefore, the space cost of all entries is $O((nodeSum + N)(d + |D_c|))$.

The other part of space cost is the inverted indexes corresponding to nodes. Note that, in the inverted lists, one term is at most related to one tuple for each entry that represents an image object, and is at most related to $c$ tuples for each entry that represents a node. In addition, the total number of terms is no more than $|D_c|$. Thus, the total space cost of all inverted indexes is $O((nodeSum \times c + N)|D_c|)$.

To sum up, the space cost of an MHIM-tree is

$$O((nodeSum + N)d + (nodeSum(c + 1) + 2N)|D_c|).$$

Note that, since the maximum number of nodes is $\frac{N-1}{m-1}$ (stated in the proof of Theorem 2), we can relax the above result, obtaining the following one by replacing *nodeSum*: $O(Nd + (\frac{c}{m} + 2)N|D_c|)$. □

**Theorem 4**. *For a top-k query Q on an MHIM-tree containing N image objects, the time complexity in the best case is* $O((k + \lceil\log_M \frac{N}{k}\rceil)(d + |Q_k|))$, *and in the worst case is* $O(N(d + |Q_k|))$.

*Proof.* The time cost for visiting a node or an image object is $O(d + |Q_k|)$, according to (4) and (8). In the best case, all image objects of the query result are close to one another in the MHIM-tree. Therefore, in the retrieval process, the number of visited nodes and image objects is at least

$$\begin{aligned} & k + (\lceil\frac{k}{M}\rceil + \lceil\frac{k}{M^2}\rceil + \cdots + 1) + \\ & (\lceil\log_M N\rceil - \lceil\log_M k\rceil) \\ = {} & k + \frac{k-1}{M-1} + \lceil\log_M \frac{N}{k}\rceil \approx O(k + \lceil\log_M \frac{N}{k}\rceil). \end{aligned}$$

Thus, the time complexity is $O((k + \lceil\log_M \frac{N}{k}\rceil)(d + |Q_k|))$. In the worst case, we may need to visit the whole tree in the retrieval, thereby the number of visited nodes and image objects is at most

$$N + maxNode = N + \frac{N-1}{m-1} \approx O(N).$$

Thus, the time complexity is $O(N(d + |Q_k|))$. □

## 8 Experiments

### 8.1 Experimental Settings

*Datasets.* In our experiments, we use three benchmark datasets: IAPR[42], LabelMe[43] and NUS-WIDE[44]. NUS-WIDE is a large social image dataset to test the scalability of our method, used to investigate if our method is promising for processing large-scale image retrieval. For ease of reference, Table 2 summarizes the main properties of these three datasets. In order to enhance the reliability and comparability of experimental results, for IAPR and LableMe we scale all images to the same size, and extract 512-dimensional GIST descriptor for each image as the visual vector. For NUS-WIDE, we use 500-dimensional bag-of-words for each image (included in the dataset) as the visual vector.

**Table 2.** Properties of Dataset

| Dataset | Images | Distinct Words | Words | Words per Image |
|---|---|---|---|---|
| IAPR | 20 000 | 7 873 | 348 630 | 1–55 |
| LabelMe | 73 000 | 19 291 | 442 215 | 1–317 |
| NUS-WIDE | 269 648 | 425 000 | 4 949 317 | 1–632 |

*Evaluation Metrics.* In our experiments, four evaluation metrics are used: 1) the construction time of our MHIM-tree; 2) the runtime of executing top-$k$ image retrieval; 3) the I/O cost per query; 4) the mean average precision (MAP), which is to measure the retrieval quality. Note that, we adopt 60 query topics and the ground truth for top-1 000 retrieval task used in ImageCLEF2007[37] to evaluate the accuracy for the IAPR dataset. For LabelMe and NUS-WIDE, there is no ground truth for top-$k$ image retrieval, thereby we randomly choose 1 000 image objects (with image and textual caption) to form the query set, and only evaluate the running efficiency.

*Paremeter Settings.* Following [36], we set $\lambda = 0.2$ in our experiments. In addition, the parameter $k$ is set to $\{1, 10, 100, 1\,000\}$, where 1 000 is the default value. To evaluate the influence of the dimension in the hashing process, we set $d \in \{32, 64, 128, 256\}$, in which $d = 128$ is the default setting. To investigate the impact of the fanout (branching factor $B$) of our MHIM-tree, we set $B \in \{50, 100, 200, 300, 400, 600\}$ for the IAPR dataset, in which $B = 400$ is the default setting. For the LabelMe dataset, the values we use are $\{100, 200, 400, 600, 800, 1\,200\}$, and $B = 800$ is set to the default value. For the NUS-WIDE dataset, we use $B = 1\,200$ as the default value. A major reason we use different $B$ for these datasets is that the sizes of these datasets are different, and more micromesh settings allow us to find exactly the impact trend and help us find appropriate default values for other tests. Also, we study the impact of the parameter $\alpha$, which is used to adjust the weight between textual and visual parts when calculating the similarity score. Specifically, we set $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, and the default value is 0.5. Experiments are executed on an Intel® Core™ i5-5200U CPU @2.20 Hz and 4 GB RAM.

*Compared Methods.* For our problem, it is challenging to find an appropriate baseline for comparison, although there have been some representative studies[2,18–21,28] that investigate the content-and-text based image retrieval (CTBIR). Among these studies, most of them are devoted to designing better fusion techniques (e.g., [18–21]). That is to say, they focused on how to effectively combine text and content information to improve the retrieval quality. Our paper directly uses their fusion techniques as mentioned in Section 3. In fact, it is unclear on how to perform top-$k$ image retrieval for these papers. These studies are essentially orthogonal to our work. Besides, the work in [2] focused on developing advanced hashing techniques to improve the retrieval quality; for this paper it is also unclear on how to perform top-$k$ image retrieval. Our paper directly uses the existing hashing technique. Note that, as we discussed in Subsection 6.1, designing more effective hashing techniques is not the focus of this paper. Thus, our work is also orthogonal to [2]. Among all these studies, the paper[28] may be the closest to our work, since it mentions the retrieval efficiency by directly exploiting an existing technique "inverted index". Thus, we would like to choose their method as Baseline1. Note that this paper uses visual words to compute the similarity score, which is different from our similarity score measurement, and it could incur some deviations in evaluation. For the fair comparison, we adapt this method by using also the inverted index and the same similarity measurement, getting a version similar to this method. The general steps of the baseline are briefly described as follows.

It first creates an inverted index for all image objects, which records $p(t|I_t)$. For a top-$k$ query, it uses the inverted index to obtain the text relevancy between the query and all image objects by (2), and then sorts all image objects in the descending order to form an array. After that, it traverses the array, and calculates the similarity score for each visited image object. During the traversal, it keeps a priority queue to record $k$

objects with the highest similarity scores. When visiting an image object $I$ whose text relevancy is lower than a threshold $\tau$, it stops traversing and returns all image objects in the queue as the query result, where $\tau = \frac{S_{\min}-\alpha}{1-\alpha}$, and $S_{\min}$ refers to the minimum similarity score in the priority queue.

Furthermore, we adopt Manhattan hashing to optimize Baseline1 according to [22], and then get the Baseline2 method. We will compare our method with Baseline1 and Baseline2, to enhance the comparability and reliability of our experiments.

## 8.2 Experimental Results of CATIRI

*Indexing Construction Cost.* Fig.2 shows the construction time of MHIM-tree, and it is easy to find that the construction time is almost proportional to the fanout for all datasets. In addition, the higher the dimension is, the longer the construction time is. This is consistent to our theoretical analysis in Section 7.

*Query Time.* Fig.3 shows the query time of MHIM-tree to perform the top-$k$ image retrieval with different fanouts. It can be seen that the runtime is almost proportional to dimension $d$. The smaller $d$ is, the less the query time is. Besides, one can see that, when the fanout $B$ is very small, the runtime is long and decreases rapidly when $B$ increases. This implies that a too small fanout value could hurt the query performance. When the fanout reaches a certain value, the runtime performance enters a steady stage. Nevertheless, when the fanout increases to a too large value, the performance starts to degrade. This implies that a too larger fanout value could be not very helpful. This is because, by limiting the capacity of each node, the fanout directly determines the height of MHIM-tree. Clearly, when the fanout is too small, the index tree is so high that the retrieval needs a long time to travel. On the other hand,

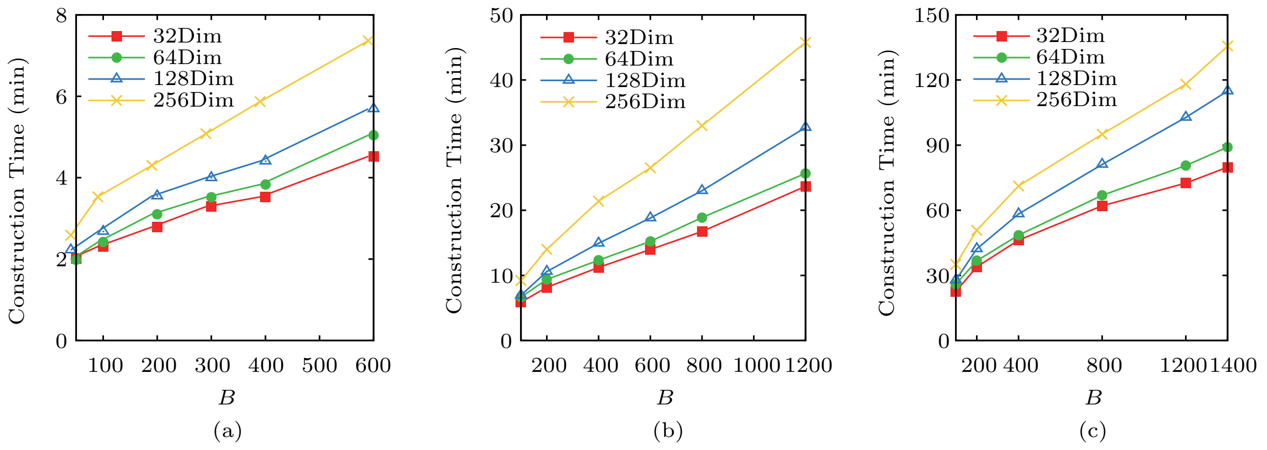


Fig.2. Construction time vs $B$. (a) IAPR. (b) LabelMe. (c) NUS-WIDE.

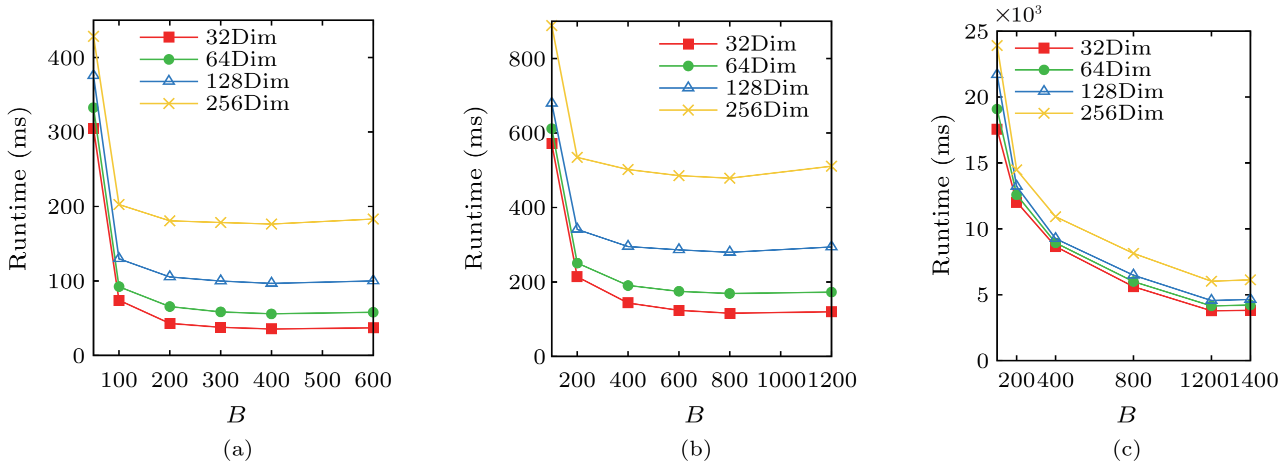


Fig.3. Runtime vs $B$. (a) IAPR. (b) LabelMe. (c) NUS-WIDE.

when the fanout is too large, the height of the MHIM-tree is so small that it offers little help for the retrieval. This essentially shows us that it is very important for the MHIM-tree to choose an appropriate fanout.

Fig.4 shows the runtime when we vary $\alpha$. It can be seen that there is no obvious fluctuation in terms of runtime. This implies that $\alpha$ has little effect on the runtime. On the other hand, as we expected, the runtime is still almost proportional to the dimension $d$.

The query runtime performance for different $k$ is shown in Fig.5. On the one hand, the runtime increases when $k$ is relatively small, while the growth speed turns slow when $k$ reaches a large enough value. The reason is same to our analysis for I/O performance. It proves that the pruning ability of our MHIM-tree is powerful and scalable. On the other hand, one can see that the dimension $d$ can make impact on the performance, and usually the larger the dimension is, the longer the runtime is. This is because computing the distances in the retrieval needs to consider each dimension.

*I/O Cost.* Fig.6 shows the I/O cost per query for different fanouts. One can see that the I/O cost decreases when the fanout increases. This is mainly because both the number of nodes and the height of the MHIM-tree decrease when the fanout $B$ increases. In addition, as we expected, the dimension $d$ rarely influences the I/O cost.

Fig.7 shows the impact of $k$ on the I/O cost. We can see that when $k$ is large enough, the I/O cost will not increase when $k$ increases. The main reason is that, the pruning power of the MHIM-tree plays a greater role in the retrieval when $k$ is large. In addition, we can also see that, for different dimensions, the I/O cost is almost equivalent. This result is consistent to our previous discussion.

*Accuracy.* Furthermore, we also study the impact of $B$ on the retrieval quality. From Fig.8(a) we can see that, the fanout has no influence on MAP. This fur-

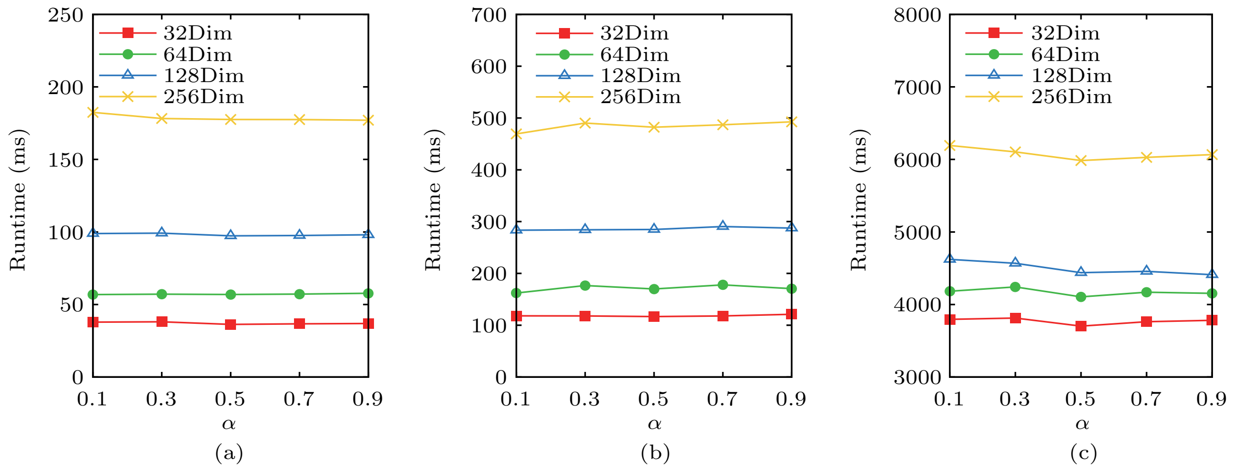


Fig.4. Runtime vs $\alpha$. (a) IAPR. (b) LabelMe. (c) NUS-WIDE.

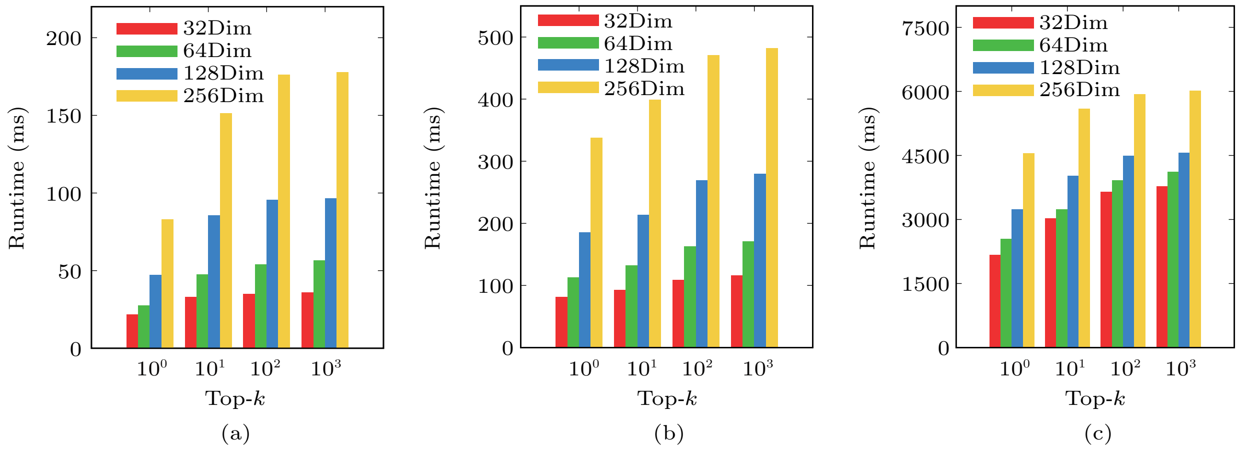


Fig.5. Runtime vs $k$. (a) IAPR. (b) LabelMe. (c) NUS-WIDE.

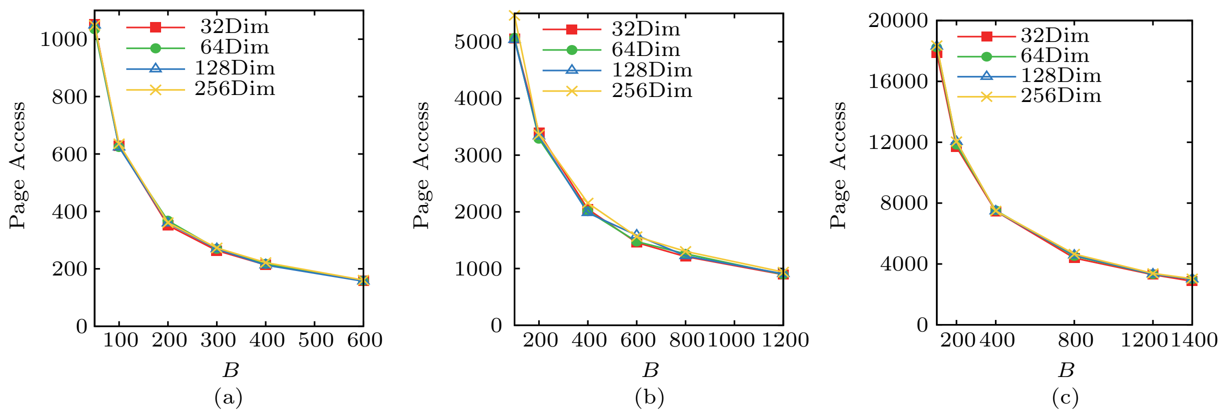


Fig.6. I/O cost vs $B$. (a) IAPR. (b) LabelMe. (c) NUS-WIDE.

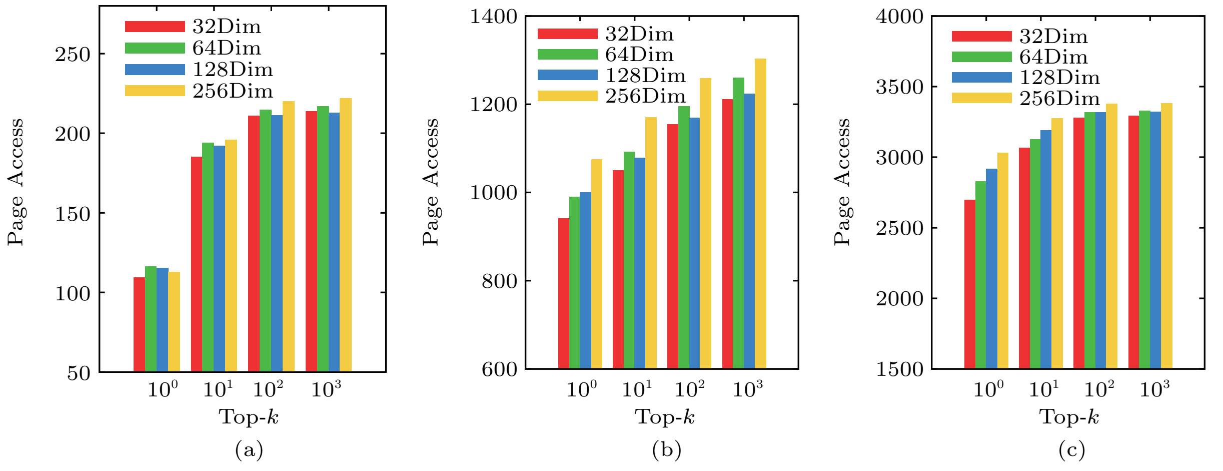


Fig.7. I/O cost vs $k$. (a) IAPR. (b) LabelMe. (c) NUS-WIDE.

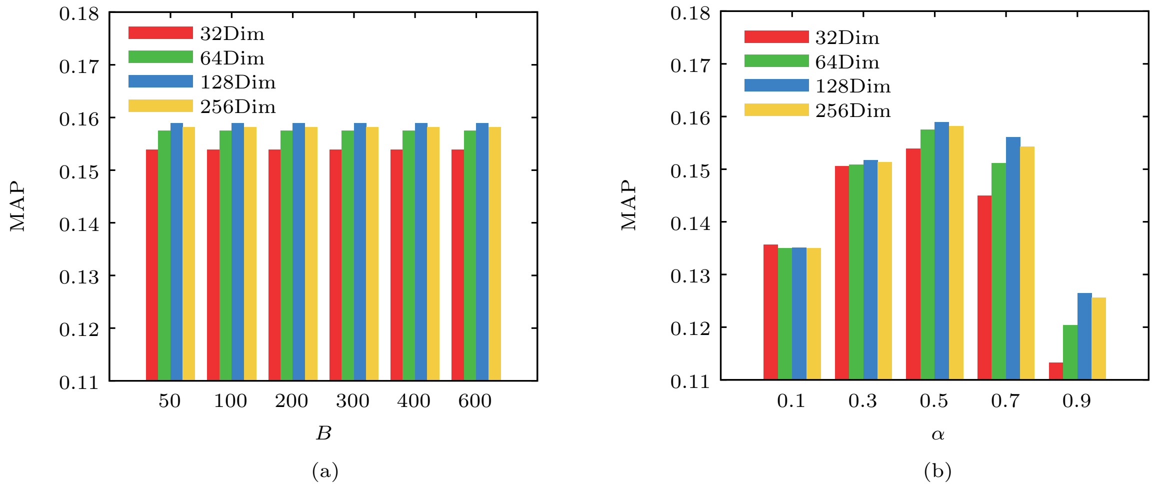


Fig.8. Accuracy. (a) MAP vs $B$. (b) MAP vs $\alpha$.

ther verifies that the fanout $B$ is essentially irrelevant to the similarity score. The MAP values for different $\alpha$ are shown in Fig.8(b). We can see that the MAP first ascends and then descends with the increase of $\alpha$, and reaches the peak at $\alpha = 0.5$. These results validate that the fusion of visual and text information can essentially improve the accuracy of query, and the best performance can be achieved at the balance point of the visual and the text information. Another phenomenon is that 128 dimensions (128Dim) achieve the best MAP for all cases except $\alpha = 0.1$. MAP keeps nearly unchanged for $\alpha = 0.1$, because $\alpha$ is so small that the visual similarity makes almost no contribution to the query result. In addition, we can see that the most appropriate dimension for our experiments could be 128.

### 8.3 Comparison with Baselines

*Query Quality.* Table 3 shows the MAP values of our method and the baselines on the IAPR dataset. It can be seen that the MAP of our method is close to that of the baselines. This means that our method can successfully preserve the retrieval accuracy. The major reason is that retrieval accuracy mainly depends on the fusion technique and hashing method, thereby CATIRI directly uses existing fusion technique and hashing technique, obtaining similar retrieval quality.

**Table 3.** MAP vs $\alpha$ of Different Methods

| Method | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| CATIRI | 0.135 1 | 0.151 7 | 0.158 9 | 0.156 1 | 0.126 5 |
| Baseline1 | 0.132 1 | 0.139 8 | 0.149 9 | 0.155 8 | 0.115 9 |
| Baseline2 | 0.137 4 | 0.153 3 | 0.155 9 | 0.152 6 | 0.126 4 |

*Runtime.* Fig.9 and Fig.10 show the runtime time of our method and the baselines. It is easy to find that our MHIM-tree significantly outperforms the baseline algorithms under all settings of $\alpha$ and $k$. And our method almost outperforms both baselines by an order of mag-

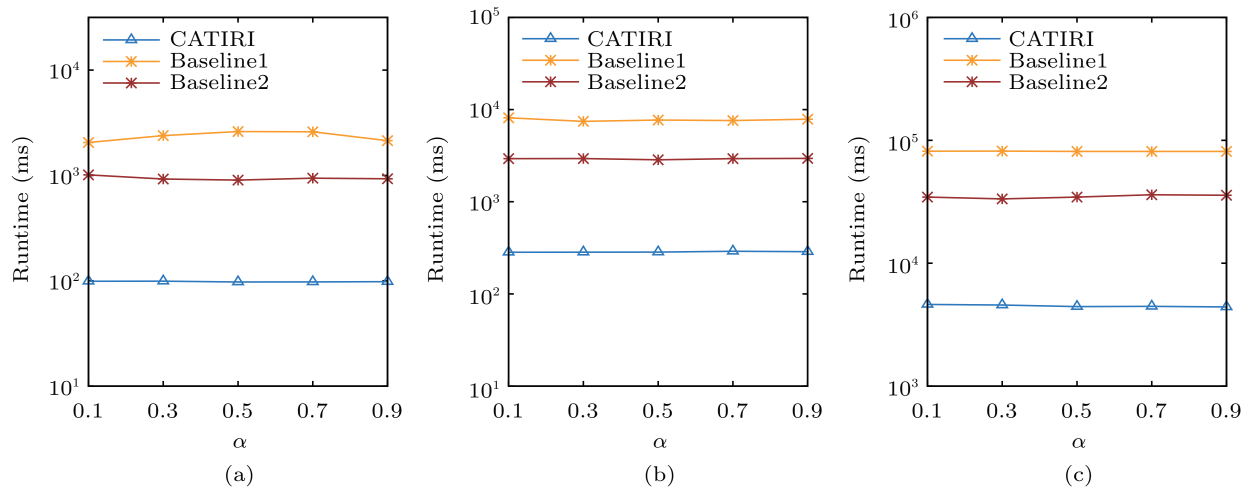


Fig.9. Runtime vs $\alpha$ of different methods. (a) IAPR. (b) LabelMe. (c) NUS-WIDE.

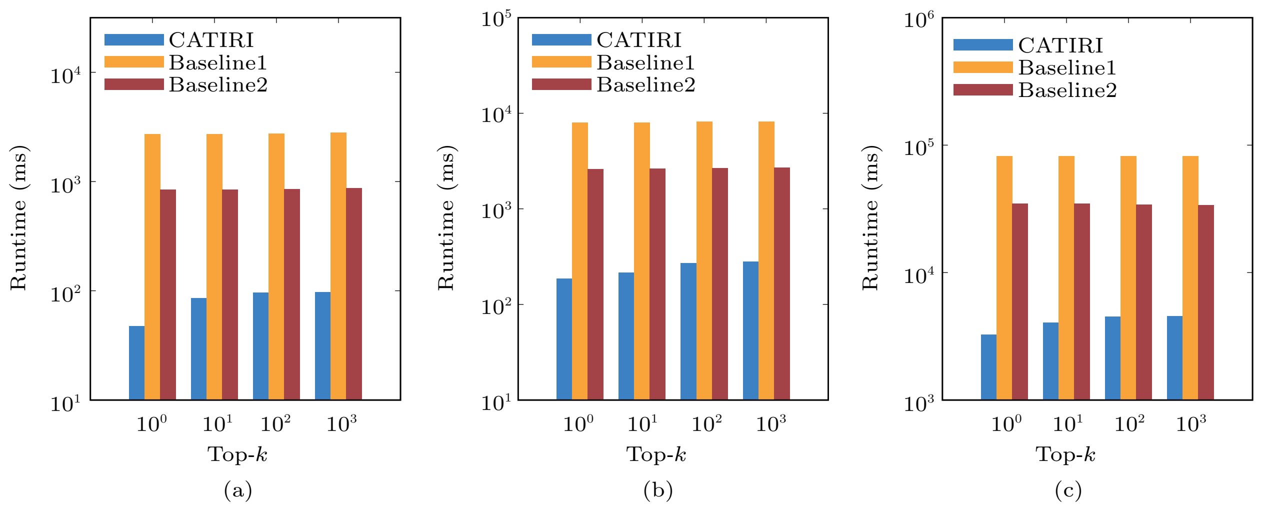


Fig.10. Runtime vs $k$ of different methods. (a) IAPR. (b) LabelMe. (c) NUS-WIDE.

nitude. This essentially demonstrates the effectiveness and efficiency of our proposed method.

*Construction Cost.* Table 4 shows the index construction cost of our method and baselines. It can be seen that the construction time of our method is larger than that of the baselines, but it is within the acceptable range. Considering the huge improvement in retrieval efficiency of our method, it is worth sacrificing a little construction time to get better retrieval efficiency.

**Table 4.** Construction Time (Minute) of Different Methods

| Method | IAPR | LabelMe | NUS-WIDE |
|---|---|---|---|
| CATIRI | 4.41 | 23.00 | 102.77 |
| Baseline1 | 0.42 | 0.70 | 5.62 |
| Baseline2 | 1.62 | 6.00 | 25.25 |

## 9 Conclusions

In this paper we proposed CATIRI for the content-and-text based image retrieval. CATIRI consists of three phases. It first transforms the image query problem into the one similar to spatial-keyword query by feature hashing, then builds an MHIM-tree to manage the textual and visual information, and finally performs the retrieval based on our top-$k$ query algorithm that fully exploits the MHIM-tree and a set of important properties. We presented theoretical analysis for CATIRI, and experimentally demonstrated that it can remarkably improve retrieval efficiency while preserving the quality of retrieval results.

## References

[1] Datta R, Joshi D, Li J, Wang J Z. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 2008, 40(2): Article No. 5.

[2] Long M, Cao Y, Wang J, Yu P S. Composite correlation quantization for efficient multimodal retrieval. In *Proc. the 39th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Jul. 2016, pp.579-588.

[3] Zhu L, Shen J, Xie L, Cheng Z. Unsupervised visual hashing with semantic assistant for content-based image retrieval. *IEEE Trans. Knowledge and Data Engineering*, 2017, 29(2): 472-486.

[4] Xu B, Bu J, Chen C, Cai D, He X. EMR: A scalable graph-based ranking model for content-based image retrieval. *IEEE Trans. Knowledge and Data Engineering*, 2015, 27(1): 102-114.

[5] Shen H T, Jiang S, Tan K L, Huang Z, Zhou X. Speed up interactive image retrieval. *The VLDB Journal*, 2009, 18(1): 329-343.

[6] Falchi F, Lucchese C, Orlando S, Perego R, Rabitti F. Caching content-based queries for robust and efficient image retrieval. In *Proc. the 12th Int. Conf. Extending Database Technology: Advances in Database Technology*, Mar. 2009, pp.780-790.

[7] Zhang C, Chai J Y, Jin R. User term feedback in interactive text-based image retrieval. In *Proc. the 28th Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Aug. 2005, pp.51-58.

[8] Li W, Duan L, Xu D, Tsang I W. Text-based image retrieval using progressive multi-instance learning. In *Proc. Int. Conf. Computer Vision*, Nov. 2011, pp.2049-2055.

[9] Wu L, Jin R, Jain A K. Tag completion for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2013, 35(3): 716-727.

[10] Tong S, Chang E. Support vector machine active learning for image retrieval. In *Proc. the 9th ACM Int. Conf. Multimedia*, Sept. 2001, pp.107-118.

[11] Liu D, Hua K A, Vu K. Fast query point movement techniques with relevance feedback for content-based image retrieval. In *Proc. the 10th Int. Conf. Extending Database Technology*, Mar. 2006, pp.700-717.

[12] Kulis B, Grauman K. Kernelized locality-sensitive hashing for scalable image search. In *Proc. the 12th IEEE Int. Conf. Computer Vision*, Sept. 2009, pp.2130-2137.

[13] Smeulders A W M, Worring M, Santini S, Gupta A, Jain R C. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2000, 22(12): 1349-1380.

[14] Deng J, Berg A C, Li F F. Hierarchical semantic indexing for large scale image retrieval. In *Proc. the 24th IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2011, pp.785-792.

[15] Ooi B C, Tan K L, Chua T S, Hsu W. Fast image retrieval using color-spatial information. *The VLDB Journal*, 1998, 7(2): 115-128.

[16] Xia H, Wu P, Hoi S C H, Jin R. Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In *Proc. the 35th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Aug. 2012, pp.55-64.

[17] Christel M G. Examining user interactions with video retrieval systems. In *Proc. the 2017 International Society for Optical Engineering*, Oct. 2007, Article No. 650606.

[18] Zhou X S, Huang T S. Unifying keywords and visual contents in image retrieval. *IEEE Multimedia*, 2002, 9(2): 23-33.

[19] Zagoris K, Chatzichristofis S A, Arampatzis A. Bag-of-visual-words vs global image descriptors on two-stage multimodal retrieval. In *Proc. the 34th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Dec. 2011, pp.1251-1252.

[20] Caicedo J C, Moreno J G, Niño E A, González F A. Combining visual features and text data for medical image retrieval using latent semantic kernels. In *Proc. the 11th ACM SIGMM Int. Conf. Multimedia Information Retrieval*, Mar. 2010, pp.359-366.

[21] Clinchant S, Ah-Pine J, Csurka G. Semantic combination of textual and visual information in multimedia retrieval. In *Proc. the 1st ACM Int. Conf. Multimedia Retrieval*, Apr. 2011, Article No. 44.

[22] Kong W, Li W J, Guo M. Manhattan hashing for large-scale image retrieval. In *Proc. the 35th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Aug. 2012, pp.45-54.

[23] Zobel J, Moffat A. Inverted files for text search engines. *ACM Computing Surveys*, 2006, 38(2): Article No. 6.

[24] Ciaccia P, Patella M, Zezula P. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. the 23rd Int. Conf. Very Large Data Bases*, Aug. 1997, pp.426-435.

[25] Rasiwasia N, Pereira C J, Coviello E, Doyle G, Lanckriet G R G, Levy R, Vasconcelos N. A new approach to cross-modal multimedia retrieval. In *Proc. the 18th ACM Int. Conf. Multimedia*, Oct. 2010, pp.251-260.

[26] Yang C, Lozano-Pérez T. Image database retrieval with multiple-instance learning techniques. In *Proc. the 16th Int. Conf. Data Engineering*, Feb. 2000, pp.233-243.

[27] Natsev A, Rastogi R, Shim K. WALRUS: A similarity retrieval algorithm for image databases. In *Proc. the 1999 ACM SIGMOD International Conference on Management of Data*, Jun. 1999, pp.395-406.

[28] Mamou J, Mass Y, Shmueli-Scheuer M, Sznajder B. A unified inverted index for an efficient image and text retrieval. In *Proc. the 32nd Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Jul. 2009, pp.814-815.

[29] Rabitti F, Savino P. An information retrieval approach for image databases. In *Proc. the 18th Int. Conf. Very Large Data Bases*, Aug. 1992, pp.574-584.

[30] Chu W W, Ieong I T, Taira R K. A semantic modeling approach for image retrieval by content. *The VLDB Journal*, 1994, 3(4): 445-477.

[31] Brown L, Gruenwald L. A prototype content-based retrieval system that uses virtual images to save space. In *Proc. the 27th Int. Conf. Very Large Data Bases*, Sept. 2001, pp.693-694.

[32] Chen L, Gao Y, Xing Z, Jensen C S, Chen G. I2RS: A distributed geo-textual image retrieval and recommendation system. *Proceedings of the VLDB Endowment*, 2015, 8(12): 1884-1887.

[33] Oliva A, Torralba A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. Journal of Computer Vision*, 2001, 42(3): 145-175.

[34] Sivic J, Zisserman A. Video Google: A text retrieval approach to object matching in videos. In *Proc. the 9th IEEE Int. Conf. Computer Vision*, Oct. 2003, pp.1470-1477.

[35] Ponte J M, Croft W B. A language modeling approach to information retrieval. In *Proc. the 21st Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Aug. 1998, pp.275-281.

[36] Zhai C, Lafferty J. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Information Systems*, 2004, 22(2): 179-214.

[37] Depeursinge A, Müller H. Fusion techniques for combining textual and visual information retrieval. In *ImageCLEF, Experimental Evaluation in Visual Information Retrieval*, Müller H, Clough P, Deselaers T, Caputo B (eds.), Springer, 2010, pp.95-114.

[38] Wang J, Liu W, Kumar S, Chang S. Learning to hash for indexing big data — A survey. *Proceedings of the IEEE*, 2016, 104(1): 34-57.

[39] Cao X, Chen L, Cong G, Jensen C S, Qu Q, Skovsgaard A, Wu D, Yiu M L. Spatial keyword querying. In *Proc. the 31st Int. Conf. Conceptual Modeling*, Oct. 2012, pp.16-29.

[40] Gong Y, Lazebnik S, Gordo A, Perronnin F. Iterative quantization: A procrustean approach to learning binary codes. In *Proc. the 24th IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2011, pp.817-824.

[41] Hjaltason G R, Samet H. Distance browsing in spatial databases. *ACM Trans. Database Systems*, 1999, 24(2): 265-318.

[42] Grubinger M, Clough P, Müller H, Deselaers T. The IAPR TC-12 benchmark: A new evaluation resource for visual information systems. In *Proc. International Conference on Language Resources and Evaluation*, May 2006, pp.13-23.

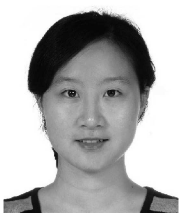
[43] Russell B C, Torralba A, Murphy K P, Freeman W T. LabelMe: A database and web-based tool for image annotation. *Int. Journal of Computer Vision*, 2008, 77(1/2/3): 157-173.

[44] Chua T S, Tang J, Hong R, Li H, Luo Z, Zheng T Y. NUS-WIDE: A real-world web image database from National University of Singapore. In *Proc. the 8th ACM Int. Conf. Image and Video Retrieval*, Jul. 2009, Article No. 48.

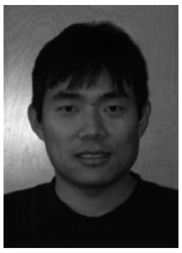
**Mengqi Zeng** is working toward her Master's degree in the Department of Computer Science and Engineering at Shanghai Jiao Tong University, Shanghai. Her research interests include information retrieval, database, and distributed computing.

**Bin Yao** is an associate professor in the Department of Computer Science and Engineering at Shanghai Jiao Tong University, Shanghai. He obtained his Ph.D. degree in computer science from the Department of Computer Science at Florida State University, Tallahassee, in 2011. His research interests include management and indexing of large databases, query processing in spatial and multimedia databases, string and keyword search, and scalable data analytics.

**Zhi-Jie Wang** received his Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai. He is currently a research associate professor at the School of Data and Computer Science, Sun Yat-sen University (SYSU), Guangzhou. Before joining SYSU, he was a postdoctoral research fellow at the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong. His current research interests include data processing and analysis for spatial/temporal data, video/image data, etc. He is a member of CCF and ACM.



**Yanyan Shen** is a tenure-track assistant professor in the Department of Computer Science and Engineering at Shanghai Jiao Tong University, Shanghai. She obtained her Ph.D. degree from National University of Singapore, Singapore, in 2015. Her research interests include data-driven machine learning, distributed data processing, and data integration and usability.



**Feifei Li** received his B.S. degree in computer engineering from the Nanyang Technological University, Singapore, in 2002 and his Ph.D. degree in computer science from the Boston University, Boston, in 2007. He is currently an professor in the School of Computing, University of Utah, Salt Lake City. His research interests include database and data management systems and big data analytics.

**Jianfeng Zhang** is currently the chief technology officer (CTO) and leader of strategy group in Alibaba Group, Hangzhou. He joined Alibaba Group in 2004.

**Hao Lin** is currently a researcher in the Department of Platform Architecture, Alibaba Group, Hangzhou. He joined Alibaba Group in 2007.

**Minyi Guo** is the head of Department of Computer Science and Engineering, and the director of Embedded and Pervasive Computing Center at Shanghai Jiao Tong University, Shanghai. He received his Ph.D. degree in information science from University of Tsukuba, Tsukuba, in 1998. His research interests include parallel and distributed processing, parallelizing compilers, cloud computing, pervasive computing, software engineering, embedded systems, green computing, and wireless sensor networks. He is currently a fellow of CCF and IEEE and a member of ACM.