Computer Science and Information Systems 16(2):615–638 https://doi.org/10.2298/CSIS181010012

On Approximate *k*-Nearest Neighbor Searches Based on the Earth Mover's Distance for Efficient Content-Based Multimedia Information Retrieval*

Min-Hee Jang[1], Sang-Wook Kim[2], Woong-Kee Loh[3†], and Jung-Im Won[4†]

1 Mobile Communication Division, Samsung Electronics Co, Korea
zzmini@agape.hanyang.ac.kr
2 Department of Electronics and Computer Engineering Hanyang University, Korea
wook@hanyang.ac.kr
3† Department of Software, Gachon University, Korea
wkloh2@gachon.ac.kr
4† Smart Computing Lab. Hallym University, Korea
jiwon@hallym.ac.kr

Abstract. The Earth Mover's Distance (EMD) is one of the most-widely used distance functions to measure the similarity between two multimedia objects. While providing good search results, the EMD is too much time consuming to be used in large multimedia databases. To solve the problem, we propose an approximate k-nearest neighbor (k-NN) search method based on the EMD. In the proposed method, the overhead for both disk accesses and EMD computations is reduced significantly, thanks to the approximation. First, the proposed method builds an index using the M-tree, a distance-based multi-dimensional index structure, to reduce the disk access overhead. When building the index, we reduce the number of features in the multimedia objects through dimensionality-reduction. When performing the k-NN search on the M-tree, we find a small set of candidates from the disk using the index and then perform the post-processing on them. Second, the proposed method uses the approximate EMD for index retrieval and post-processing to reduce the computational overhead of the EMD. To compensate the errors due to the approximation, the method provides a way of accuracy improvement of the approximate EMD. We performed extensive experiments to show the efficiency of the proposed method. As a result, the method achieves significant improvement in performance with only small errors: the proposed method outperforms the previous method by up to 67.3% with only 3.5% error.

Keywords: Earth mover's distance, content-based information retrieval, k-nearest neighbor query.

---

*This is an extended version of a WIMS'18 conference paper.

† Co-Corresponding authors: Jung-Im Won (Hallym University), Email address: jiwon@hallym.ac.kr
Woong-Kee Loh (Gachon University), Email address: wkloh2@gachon.ac.kr

1. Introduction

Due to the recent advances in digital technologies, an enormous number of multimedia objects are now available over the Internet. In order to find the specific objects needed by the users from such a huge multimedia pool, an efficient search technique is highly essential [1], [2], [3], [4], [28]. Content-based information retrieval (CBIR) is the search technique based on the contents of multimedia objects. It finds the multimedia objects similar to the given query object based on features such as color distribution, shape, and texture [1], [5], [6], [26]. Since the CBIR is given with a multimedia object as a query example, it is more intuitive than non-CBIR such as keyword-based retrieval [1].

There are two types of the CBIR, namely the range query and the *k*-nearest neighbor (*k*-NN) query [7], [8], [9], [10], [27]. The range query finds a set of objects whose distances from the query object are less than or equal to the given threshold θ. It has the weakness that it is difficult to find a proper θ, thereby returning too big or too small search results according to θ. In the worst case, the range query may return either an empty or the whole dataset [9], [11], [12]. The *k*-NN query finds *k* objects that are nearer from a query object than other objects. Since the *k*-NN query does not require a hard-to-estimate distance θ from the query, users can converge on their wanting objects more quickly. In this paper, we focus our attention on the *k*-NN CBIR query.

For the CBIR, a multimedia object can be represented as an *n*-dimensional probabilistic histogram [8], [12], [13]. An *n*-dimensional histogram *H* is composed of *n* bins, where each bin represents a probability p_i, i.e., $H = \{p_1, p_2, ..., p_n\}$. The probabilities p_i can be located in a multi-dimensional space rather than serially aligned. For example, an image with 10 colors in the *RGB* space is represented as a 10-dimensional histogram, i.e., $n = 10$. The weight of each color in the image is represented as the probability of the corresponding bin in the histogram, and the red (*R*), green (*G*), and blue (*B*) components of each color are located in the three-dimensional space.

For the CBIR on multimedia databases, we need a distance function which measures the similarity between two objects. There have been a number of distance functions proposed such as the Euclidean distance or the $\chi 2$ statistic. While they can be computed very fast and give good results in some cases, they do not take into account all possible variations of matching, which could cause inaccurate search results [14]. The Earth mover's distance (EMD) is a representative distance function for the CBIR [8], [11], [12], [13] to address this problem [14], [15]. The EMD between two histograms is defined as the minimum work needed to transform one histogram into the other by moving portions of bins [14]. The work is defined as the weight (portion) of a bin moved times the moving distance. Due to the high quality of the search results based on the EMD, it is adopted in various applications [14], [15]. However, the time complexity of EMD computation is $O(n^3 \log n)$, where *n* is the number of bins in a histogram, and thus the EMD cause significant overhead for large multimedia databases [8], [12], [14].

Many efforts have been made for efficient EMD-based CBIR: approximation methods [15], [16], lower-bound methods [17], [18], and indexing methods [8], [12]. The approximation methods compute the approximate EMD very quickly. They improved the EMD computation time in an order of magnitude; however, they have scalability problems that incur large disk access overhead. The lower-bound methods first find a set of candidates using lower-bound functions and then compute the actual EMD between the candidates and the query object to find the real *k*-NN results. They significantly reduce the number of the EMD computations owing to the lower-bound

functions; but they also have scalability problems. The indexing methods are proposed to reduce both the disk access overhead and the computational overhead. They first find a set of candidates using the indexes and then compute the actual EMD between the candidates and the query to return the final *k*-NN result. By using the indexes, they can reduce the disk access overhead. However, they still require a considerable number of costly EMD computations in the post-processing step, because the number of candidates is much larger than *k*. Therefore, for efficient EMD-based CBIR, both the disk access overhead and the EMD computation overhead should be reduced at the same time.

In this paper, we propose an efficient approximate *k*-NN method based on the EMD. The proposed method uses the M-tree [7], a distance-based multi-dimensional index, to reduce the disk access overhead. The M-tree is constructed using distances between objects rather than the actual object positions in the multi-dimensional space. The M-tree generally provides good search performance for retrieving high-dimensional histograms and thus reduces the disk access overhead of the EMD-based CBIR [7], [17]. When the CBIR is done using the M-tree, it requires a large number of EMD computations between the query object and those in the index [7], [17]. Since the complexity of EMD computation is very high, it should incur a serious computational overhead and thus dramatic deterioration of the search performance. To solve the problem, we reduce the dimensionality of *n*-dimensional histogram into n' ($<< n$) through dimensionality-reduction [18] and the M-tree is constructed using the dimension-reduced histograms.

Besides, we use the approximate EMD [16] to reduce the EMD computation overhead when retrieving the M-tree. The approximation EMD, which is very close to real EMD, is computed in a linear time with a small error. Since the approximate EMD between two dimension-reduced histograms is always lower than the approximate EMD between the original histograms, there occurs no false-dismissal, which will be proven in Section 4.2. Since the M-tree is constructed using the dimension-reduced histograms, a set of candidates are obtained as the result of the M-tree search. Then, the post-processing is performed to find the real *k*-NN result by computing the EMD on original histograms of the candidates. Even in the post-processing step, a large number of EMD computations should be performed because the number of candidates is much larger than *k*. The proposed method uses the approximate EMD to speed up the post-processing. To minimize the errors due to the approximate EMD, we propose an accuracy improvement method. Extensive experiments on real datasets comparing our method with the state-of-the-art methods [8], [12], [18] were conducted to show the superiority of our method. The result reveals that our method outperforms the previous ones by up to 67.3% with the error as small as 3.5%.

This paper is organized as follows. Section 2 describes the EMD in more detail, and Section 3 briefly reviews the related work on the EMD-based CBIR and points out their weaknesses. Section 4 presents our method in detail and Section 5 evaluates its performance in comparison with existing ones. Finally, Section 6 summarizes and concludes this paper.

2. Earth Mover's Distance

Let $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$ ($\sum_{i=1}^{n} p_i = \sum_{i=1}^{n} q_i$) be two n-dimensional histograms, where p_i and q_i are bins in the histograms. A matrix $D = [d_{ij}]$ is called a ground distance matrix, where d_{ij} is the ground distance between p_i and q_j. The ground distance is defined by any standard metric, such as the Euclidean distance or the Manhattan distance [14]. A matrix $F = [f_{ij}]$ is called a flow matrix, where f_{ij} is the portion of bin (called weight or probability) transferred from p_i to q_j. The work is defined as the multiplication of flow f_{ij} and ground distance d_{ij}. The EMD between P and Q is defined as the minimum amount of work required to transform a distribution P into the other Q (or in the opposite direction) and formally written as:

$$EMD(P,\ Q) = min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} \cdot d_{ij}\right\} \quad (\mathbf{1})$$

subject to:

$$\forall i, j : f_{ij} \geq 0,$$

$$\forall i : \sum_{j=1}^{n} f_{ij} = p_i,\ and$$

$$\forall j : \sum_{i=1}^{n} f_{ij} = q_j$$

Figure 1 shows an example of computing the EMD between P and Q, which are represented as 6-dimensional histograms (i.e., $n = 6$). In the figure, the x-axis represents each bin in the histograms and the y-axis represents the weight (or probability) of each bin. When transforming P into Q, the portions of bins in P can be moved to different positions to match the bins in Q in a variety of ways. In Figure 1, the amount of work is the minimum when each bin portion in P is moved to the position designated with the same character in Q. Thus, the EMD is computed as:

$$\text{EMD}(P,Q) = 0.3 \times 2\big(Work(A)\big) + 0.3 \times 1(Work(B)) + 0.2 \times 3(Work(C)) + 0.2 \times 1(Work(D))$$

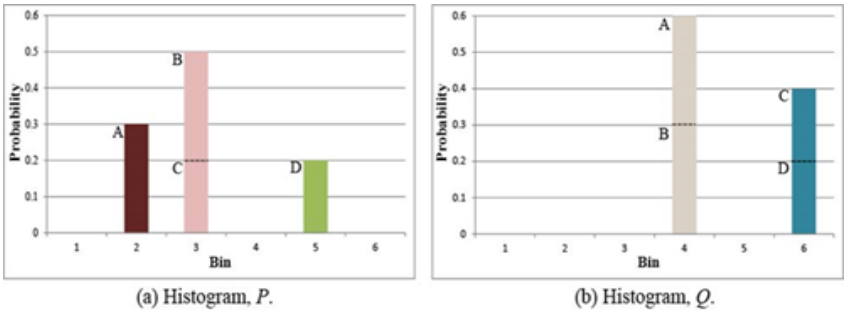


Fig. 1. Example of computing the EMD between two 6-dimensional histograms.

3. Related Work

3.1. Approximate EMD

The approximate EMD (AEMD) between two multimedia objects is computed in a time linear to the number of bins, and it is very close to the actual EMD. In a one-dimensional space, the exact EMD between two histograms is computed as follows: (1) it selects the bins at the end in the one-dimensional space from each histogram and computes the work for the common weights of the selected bins; (2) the common weights are removed from the histogram; (3) the steps (1) and (2) are repeated until all bins in two histograms are removed. While repeating the steps (1)~(3), the (partial) works are all added, and the final sum is returned as the EMD between two histograms. Since both bins are scanned only once, the time complexity of one-dimensional EMD computation is O(n) [16], which is much smaller than that of multi-dimensional EMD computation. Based on this idea, AEMD linearizes the multi-dimensional histogram using the Hilbert curve. The Hilbert curve fills the multi-dimensional space with a continuous curve as shown in figure 2 [19]. After the linearization, AEMD applies the method mentioned above.

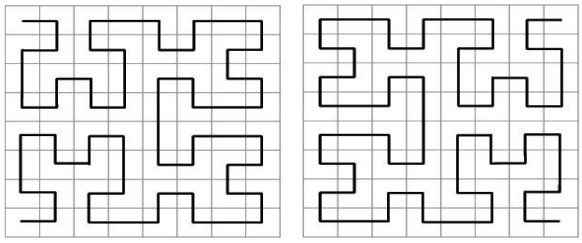


Fig. 2. Example of the Hilbert curve in two-dimensional space.

To pursue the highest accuracy, AEMD creates multiple Hilbert curves with different starting points and directions. In an m-dimensional space, there can be $(2^m \cdot m!)/2$ Hilbert curves [16]. AEMD takes the minimum of the works obtained for each of the curves. As an experimental result in [16], AEMD incurred an error up to only 4.2%, while it dramatically improved the speed by up to 34 times over the EMD. However, AEMD suffers from the scalability problem, since it should access all the objects in a multimedia database for CBIR.

3.2. Lower-bound Based on Dimensionality Reduction

As indicated in the complexity, the time for EMD computation for lower dimensional histograms should be smaller than that for higher dimensional ones. The method by Wichterich, et al. (called *LB* in this paper) transforms an n-dimensional histogram H

into an n'-dimensional one $H'(n' < n)$ using an $n \times n'$ reduction matrix $R = [r_{ij}]$ as the following equation [18]:

$$H' = H \cdot R \quad (2)$$

where it holds that:

$$\forall i,j \left(1 \leq i \leq n,\ 1 \leq j \leq n'\right),\ r_{ij} \in \{0,\ 1\},$$

$$\forall i,\ \sum_{j}^{n'} r_{ij} = 1,\ and$$

$$\forall j,\ \sum_{i}^{n} r_{ij} \geq 1$$

LB computes the ground distance between the bins in the dimensionality-reduced histograms using the *optimal reduced distance matrix* $D' = [d'_{ij'}]$. An element $d'_{ij'}$ in the matrix is defined as:

$$d'_{i'j'} = min\{d_{ij} | r_{ii'} = 1 \wedge r_{jj'} = 1\} \quad (3)$$

The optimal reduced distance matrix D' ensures that the EMD between the dimensionality-reduced histograms is always lower than (or lower-bounds) that between the original histograms [18]. The main advantages of this dimensionality reduction include efficiency, flexibility, and completeness [18]. By using this lower-bound property, LB scans all the objects in the database and returns a set of candidates. Then, LB computes the EMD between the candidate's original histogram H and the query histogram, and returns the qualified ones. This method reduces EMD's computational overhead; however, as the AEMD, it also suffers from the scalability problem, since it should access all the objects in the database.

3.3. Indexing Methods

To solve the scalability problem, a few indexing methods such as the tree-based index (called *TBI* in this paper) [8] and the normal lower-bound index (called *NLI* in this paper) [12] have been proposed. TBI employs LB^+-trees; every histogram is mapped to L domain values using the primal-dual theorem [20], and then each set of l-th ($0 \leq l < L$) values is indexed in a B^+-tree. TBI processes range queries and k-nearest queries efficiently using the trees. TBI converts the EMD-based range query into L range queries against L trees, and the objects contained in the intersection of the results obtained through the trees constitute the candidate set. Then, TBI computes the actual EMD for each candidate object to find the final query result. In TBI, the number of B^+-trees increases as the database size increases. To solve this problem, NLI employs a single Quad-tree to index multi-dimensional histograms [12]. NLI projects a histogram onto a vector and approximates it by a normal distribution. NLI then represents each normal as a point in a Hough transformed space and stores in the Quad-tree. NLI computes the lower-bound EMD in a constant time. Since NLI is based on

approximation, it still needs EMD computations in the post-processing phase on the candidates obtained through the index.

Although these methods reduce the number of EMD computations, they still require a considerable number of EMD computations in the post-processing phase. As a result of our experiment on a database composed of 3,932 high-dimensional histograms [8], [12], TBI and NLI should have performed about 350 and 300 EMD computations, respectively. The EMD computations significantly deteriorate the performance of the EMD-based CBIR. Recently, a refinement method [21] has been proposed to reduce the post-processing overhead by adopting the simplified graph incremental algorithm and the dynamic refinement order. However, the method focuses only on the post-processing and can be applied only to NLI, unlike our method explained in the next section.

4. Proposed Method

In this section, we propose an efficient approximate k-NN algorithm for EMD-based CBIR. In Section 4.1, we present a naïve approach incurring no false dismissal. In Sections 4.2 and 4.3, we refine the naïve approach for performance improvement and accuracy improvement.

4.1. Naïve Approach

The naïve approach performs the *k*-NN using the index for scalability. Since we deal with the *n*-dimensional histograms, we may consider the R-tree [22], a most widely used multi-dimensional index. However, the R-tree suffers from the so called high-dimensionality problem or high-dimensionality curse [7]: the performance of the R-tree rapidly deteriorates with the increase of the histogram dimension. If the R-tree is employed for the EMD-based CBIR, the search performance using the R-tree becomes worse than even the sequential scan when the dimension of the histograms exceeds 35 [13].

In this paper, we adopt the M-tree, a distance-based index structure [7]. While the R-tree is constructed based on the object locations in the multi-dimensional space, the M-tree is based on the distances between the objects. The distance computations for only a small number of (not all possible) object pairs are required to construct the M-tree [7]. Also, the M-tree provides good performance not only in the indexing but also in the search for high-dimensional histograms, because the M-tree performs the search using the distances previously computed when constructing the index. Figure 3 shows an example of the M-tree. Figure 3(a) shows the distribution of objects ($o_1 \sim o_{10}$) in the two-dimensional space, and Figure 3(b) shows an M-tree constructed based on distances between these objects in Figure 3(a). In the figure, we set the maximum number of entries of the internal and terminal nodes are two and three, respectively.

The naïve approach constructs the M-tree index based on the EMD between *n*-dimensional histograms and performs the *k*-NN search using the *k*-NN algorithm for the M-tree presented in [7]. The naïve approach incurs no false dismissal, since EMD satisfies the following three conditions:

(1) Positivity: $\mathrm{EMD}(P, Q) \geq 0$,
(2) Symmetry: $\mathrm{EMD}(P, Q) = EMD(Q, P)$, and
(3) Triangular inequality: $\mathrm{EMD}(P, Q) \leq \mathrm{EMD}(P, R) + \mathrm{EMD}(R, Q)$

where P, Q, and R are histograms.

However, the naïve approach has the performance problem because the EMD has the very high computation complexity $O(n^3 \log n)$ for n-dimensional histograms [14]. Since many EMD computations are required in the course of the indexing and the searching, it causes serious deterioration of overall performance of the approach. To solve this problem, we provide three extensions as follows. First, we use the dimensionality-reduction [18] explained in Section 3.2. That is, we transform n-dimensional histograms to n'-dimensional ones ($n' << n$) using the dimensionality-reduction when constructing the M-tree and performing the k-NN search. Since the dimension of histograms is reduced significantly, the EMD computation overhead should also be reduced as much. Second, we use the AEMD explained in Section 3.1 instead of the EMD [16]. The AEMD is computed in $O(n)$ time with small errors with the EMD. Third, we adopt the maximum common weight elimination to reduce the error between the AEMD and the EMD. We explain first and second extensions in Section 4.2 and third one in Section 4

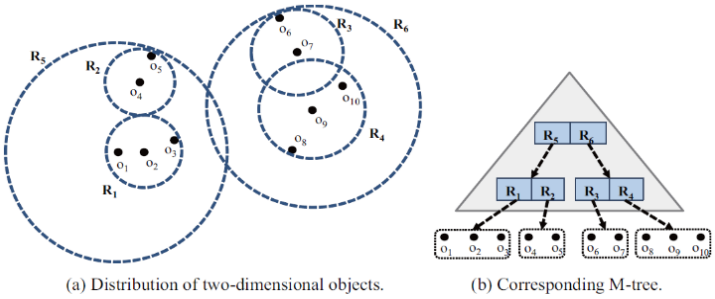


Fig. 3. A Sample M-tree.

4.2. Performance Improvement

The proposed k-NN method adopts the dimensionality-reduction [18] using the reduction matrix R for both the indexing and the searching. When indexing, we compute the EMD with the optimal reduced distance matrix D' to construct the M-tree. Likewise, when searching, we reduce the dimension of the query histogram by using same matrix R and perform the k-NN search on the M-tree with D' and the EMD. Since $n' << n$, this method performs the indexing and searching more efficiently than the naïve approach. Note that the method based on the dimensionality-reduction incurs no false dismissal. For its justification, we need the following lemma:

Lemma 1. For any two n-dimensional histograms P and Q, the following is satisfied [18]:

$$EMD_{D'}(P', Q') \leq EMD_D(P, Q) \quad \textbf{(4)}$$

where $EMD_{D'}(P', Q')$ is the EMD with the optimal reduced distance matrix D' after reducing the dimension of P and Q into P' and Q' using the reduction matrix R, and $EMD_D(P, Q)$ is the EMD with the original ground distance matrix D and the original histograms P and Q.

Proof: See [18]. □

As shown in Eq. (4), the EMD between any two dimensionality-reduced histograms lower-bounds the EMD between original histograms. Based on this lower-bounding property, we introduce a *k*-NN algorithm as shown in Figure 4.

The figure shows the *k*-NN_EMD algorithm based on the dimensionality-reduction, and the M-tree that is constructed with the dimensionality-reduced histograms in the algorithm. The *k*-NN_EMD is given a query histogram Q and the number of objects k as the input and returns an array of the *k*-NN objects as the output. In line (1), the algorithm calls *k*-NN_Search() function for the M-tree. This function is presented in [7] and performs *k*-NN search using the M-tree. Since the M-tree is constructed with the dimensionality-reduced histograms, we also reduce the dimension of the query Q into Q' and find the *k*-NN objects by invoking the *k*-NN_Search(). In line (2), we compute the original EMD, $EMD_D(Q, P)$ for each P returned from the *k*-NN_Search() and then, in line (3), we sort the result and save in the array NN. As a point, $NN[k]$ is the objects with the largest EMD from Q among all the objects in NN.

Then, we call Next-NN_Search() function with Q as the input. This function returns the nearest neighbor N next to the current nearest neighbors NN using the M-tree. For example, when we have *k*-NN objects obtained by searching the M-tree, if we call the Next-NN_Search() function, it returns the $(k+1)$-th object. If we call the Next-NN_Search() function again, it returns the $(k+2)$-th object. This function can be easily implemented using the *k*-NN_Search() function as follows. Let $k(0)$ be the predefined number of objects currently searched by the *k*-NN_Search(). When the Next-NN_Search() is called, it returns the i-th ($i \leq k(0)$) nearest neighbor among the $k(0)$ objects. If $i > k(0)$, the *k*-NN_Search() is invoked internally to search $k(1)$ ($> k(0)$) objects next farther than the $k(0)$ objects and the i-th ($i \leq k(1)$) object is returned. The Next-NN_Search() function returns the next nearest neighbor continuously by repeating this process.

In line (6), we compute $EMD_D(Q, N)$ of N and, if the result is smaller than the EMD of $NN[k]$, N is inserted into the NN preserving the order of EMD from the original Q, and the previous $NN[k]$ is discarded. We call Next-NN_Search() again in line (9). As shown in line (5), this process is repeated until $EMD_D(Q, N)$ is smaller than $EMD_{D'}(Q', NN[k]')$. If $EMD_D(Q, N)$ is larger than $EMD_{D'}(Q', NN[k]')$, we break the while loop and finally return the array NN to the user in line (11). In summary, the *k*-NN_EMD algorithm first searches for the *k*-NN candidates by using the M-tree constructed with the dimensionality-reduced histograms and then refines the *k*-NN result by comparing the actual EMD of the original high-dimensional histograms.

```
Algorithm k-NN_EMD(Q, k, NN)
Q: query histogram (Input)
k: number of neighbors (Input)
NN: array of k-nearest neighbors (Output)
1:    Call k-NN_Search(Q', k);
2:    Compute EMD_D(Q, P) of each object in k-NN
3:    Save the output to NN in the order of EMD from original Q;
4:    Call Next-NN_Search(Q) and let N be the result;
5:    while EMD_D(Q, N) ≤ EMD_D(Q', NN[k]')
6:        if EMD_D(Q, N) ≤ EMD_D(Q, NN[k])
7:            Insert N into NN preserving the order of EMD from original Q;
8:        end if
9:        Call Next-NN_Search(Q) and let N be the result;
10:   end while
11:   return NN;
```

Fig. 4. k-NN_EMD algorithm based on dimensionality reduction.

The *k*-NN_EMD guarantees no false dismissal, and it can be easily shown using Eq. (4) as follows. *NN*[*k*] is the object with the farthest EMD from *Q* in *NN* returned from the *k*-NN_EMD (in line (11)). For the next nearest neighbor *N* given by the Next-NN_Search(), it holds that EMD_D (*Q*, *NN*[*k*]) ≤ $EMD_{D'}$ (*Q'*, *N'*) according to the condition in line (5). It also holds that $EMD_{D'}$ (*Q'*, *N'*) ≤ EMD_D (*Q*, *N*) according to Eq. (4). Therefore, for any *N* in the M-tree, it holds that EMD_D (*Q*, *P*) ≤ EMD_D (*Q*, *N*) indicating that the original EMD of any object in the M-tree is larger than the original EMD of any *P* in *NN*, and thus *NN* is the set of *kNN* from *Q*.

In the *k*-NN_EMD, we use the dimensionality-reduced histograms for efficient indexing and searching; however, there is still performance deterioration even with the dimensionality-reduced histograms due to very high complexity of the EMD. Moreover, many EMD should also be computed for original histograms when refining the candidates. To solve this problem, instead of the EMD, we use the AEMD proposed by Jang et al. [16] to improve the indexing and searching efficiency. According to the experiment results in [16], the AEMD computation requires only O(*n*) time with only 4.2% average error, where the average error is defined as $\left|\frac{EMD(P,Q)-AEMD(P,Q)}{EMD(P,Q)}\right|$.

By adopting the AEMD, the proposed method performs the approximate *k*-NN search. Figure 5 shows the *k*-NN_AEMD algorithm which is a modification of the *k*-NN_EMD algorithm in Figure 4. In this algorithm, we assume the size of NN to be *k'* (> *k*).

As in the *k*-NN_EMD, the *k*-NN_AEMD is given a query histogram *Q* and the number of objects *k* as the input and returns an array *NN* of *k*-NN objects based on the EMD. The main difference of the *k*-NN_AEMD from the *k*-NN_EMD is that the *k*-NN_AEMD constructs the M-tree using the EMD but searches the tree using the AEMD. It is for reducing both the accuracy loss due to the AEMD and the computational overhead due to the EMD. If we use only the EMD as shown in Figure 4, it returns the exact *k*-NN result based on the EMD with poor search performance.

```
Algorithm k-NN_AEMD(Q, k, NN)
Q: query histogram (Input)
k: number of neighbors (Input)
NN: array of k-nearest neighbors (Output)
 1:   Compute k' as a function of k;
 2:   Call k-NN_Search(Q', k');
 3:   Compute AEMD_D (Q, P) of each object in k'-NN
 4:   Save the output to NN in the order of AEMD from original Q;
 5:   Call Next-NN_Search(Q) and let N be the result;
 6:   while AEMD_D (Q, N) ≤ AEMD_D(Q', NN[k'])
 7:        if AEMD_D(Q, N) ≤ AEMD_D(Q, NN[k'])
 8:             Insert N into NN preserving the order of AEMD from original Q;
 9:             Call Next-NN_Search(Q) and let N be the result;
10:        end if
11:   end while
12:   Post-process NN;
13:   return NN;
```

Fig. 5. k-NN_AEMD algorithm using AEMD.

On the contrary, if we use only the AEMD for both the indexing and the searching, it returns highly inaccurate results with little performance improvement over the *k*-NN_AEMD for the following reason. The search on the M-tree is based on the distances previously computed when constructing the M-tree. If we construct the M-tree using the AEMD, the distances have errors. When the M-tree is used to compute the distances between the objects in the course of *k*-NN search, the distances based on the AEMD causes additional errors. Thus, the errors are accumulated, which causes a lot of unexpected false dismissals. Therefore, we decide to use the AEMD only in the search process to minimize false dismissals.

To minimize the false dismissals due to the AEMD, we perform the *k'*-NN search instead of the *k*-NN ($k' > k$) in line (1). The *k*-NN_AEMD algorithm after the line (1) is almost the same as the *k*-NN_EMD algorithm in Figure 4 and performs the *k'*-NN search down to line (11). In line (12), the post-processing is performed to return the *k*-NN result to the user: the original EMD is computed for each object in the *k'*-NN array from the query *Q*, and *k* objects with the smallest EMD are returned in line (13). This method reduces the EMD computation overhead, because it only needs to compute the EMD for *k'* candidates in the post-processing step. There is additional computation overhead when searching *k'*-NN objects from the M-tree for *k'* larger than *k*; however, it is only fairly marginal compared with the overall performance gain obtained by the AEMD. We discuss in detail on the accuracy and the performance of the *k*-NN_AEMD algorithm in Section 5. As in the *k*-NN_EMD, the *k*-NN_AEMD incurs no false dismissal by the dimensionality-reduction. That is, the *k'*-NN result found in the *k*-NN_AEMD until the line (11) contains every object that should be eventually returned as the search result. For the justification, we need the following lemma:

Lemma 2. For any two *n*-dimensional histograms *P* and *Q*, the following is satisfied:

$$AEMD_{D'}(P', Q') \leq AEMD_D(P, Q) \quad (5)$$

where $AEMD_{D'}(P', Q')$ is the AEMD with the optimal reduced distance matrix D' after reducing the dimension of P and Q into P' and Q' using the reduction matrix R, and $AEMD_D(P, Q)$ is the AEMD with the original ground distance matrix D and the original histograms P and Q.

Proof: See the Appendix. □

We can show using Lemma 2 that no false dismissal is caused by the dimensionality-reduction in the k-NN_AEMD algorithm. This proof is almost the same as in the k-NN_EMD. $NN[k']$ is the farthest object from Q in NN obtained until the line (11) in Figure 5. For the next nearest neighbor N given by the Next-NN_Search(), it holds that $AEMD_D(Q, NN[k]) \leq AEMD_{D'}(Q', N')$ according to the condition in line (6). It also holds that $AEMD_{D'}(Q', N') \leq AEMD_D(Q, N)$ according to Eq. (5). Thus, for any object N in the M-tree, it holds that $AEMD_D(Q, P) \leq AEMD_D(Q, N)$. It indicates that the original AEMD of any object in the M-tree is larger than the original AEMD of any P in NN, and thus NN is the set of k'-NN from Q.

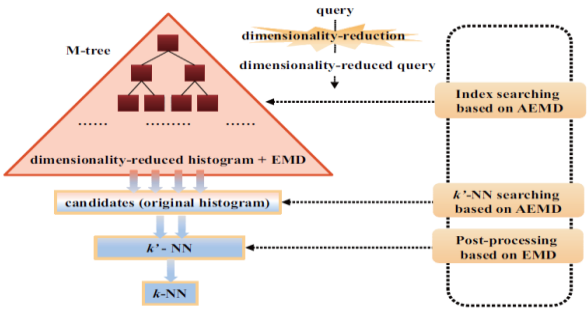


Fig. 6. Indexing and searching in the k-NN_AEMD.

Figure 6 shows the process of the k-NN_AEMD algorithm. In the figure, the M-tree is constructed based on the EMD of the dimensionality-reduced histograms. When a query Q is issued, the algorithm first reduces the dimension of the query histogram in the same manner. The algorithm finds the k'-NN objects based on the AEMD of the dimensionality-reduced histograms. Then, it finds the set of k'-NN candidates by searching the M-tree based on the AEMD of the original histograms. Finally, the original EMD for each k'-NN candidate is computed in the post-processing step, and the k-NN objects with the smallest EMD from Q are returned to the user.

4.3. Accuracy Improvement

The *k*-NN_AEMD finds k' (> k) nearest neighbors to reduce the false dismissals due to the error between the AEMD and the EMD. However, even the k'-NN search with very large k' might cause false dismissals for very high dimension histograms because the error between the AEMD and the EMD increases as the dimension increases. To reduce such error due to the AEMD, we propose the elimination of maximum common weight (MCW). The MCW is the smaller weight of the two corresponding bins with the same location *i*. For two *n*-dimensional histograms $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$, the MCW c_i ($1 \le i \le n$) is defined as $c_i = \min\{p_i, q_i\}$. The MCW-eliminated histograms P_c and Q_c are defined as follows:

$$P_c = \{p_1 - c_1, \cdots, p_n - c_n\} \text{ and } Q_c = \{q_1 - c_1, \cdots, q_n - c_n\}$$

When computing the EMD, the work of the MCW of two corresponding bins is 0, because the ground distance of the bins with the same location is 0. Since the EMD is defined as the minimum work, we get the same EMD, whether the MCW elimination is applied or not. On the contrary, when computing the AEMD, since the MCW bins might move to different locations, the work of the MCW of the two corresponding bins could be larger than 0. To solve this problem, we eliminate the MCW before the AEMD computation. That helps reduce the error between the AEMD and the EMD significantly. Our experiment result shows that the average error of the AEMD is reduced to half by applying the MCW elimination. We discuss this result in detail in Section 5.

The MCW elimination is applied in all the lines that compute the AEMD, i.e., in the lines (2) ~ (9) in the *k*-NN_AEMD algorithm in Figure 5. The lines (2), (5), (6) and (9) compute the AEMD for the n'-dimensional histograms, and the lines (3), (4), (6), (7) and (9) for the *n*-dimensional histograms. The overhead of the MCW elimination is trivial since it only needs simple arithmetic operations in the O(n) time.

Another problem that causes the error between the AEMD and the EMD is that the AEMD does not generally satisfy the triangular inequality explained in Section 4.1, while the positivity and the symmetry are satisfied. The false dismissals can be generated due to the problem in the *k*-NN_AEMD algorithm. However, our experiment result with all possible histogram pairs in a real dataset shows that only 0.2% of the pairs violate the triangular inequality. Therefore, we can claim that the false dismissals due to the violation of triangular inequality are not significant. The detail is discussed in Section 5.2.

5. Evaluation

5.1. Experimental Setup

In this section, we verify the superiority of the proposed method through extensive experiments. We used three datasets in the experiments, namely RETINA [8], [12],

[18], Shader [23], and SIMPLIcity [16], [24]. RETINA is a high-resolution image set which consists of 3,932 feline retina scans. Each image is represented as a 96-dimensional histogram, and each bin in the histogram represents a location in two-dimensional space. Shader contains 30,000 computer graphics (CG) images. Each image is represented as a 128-dimensional histogram with the bins in three-dimensional space. For each image, the weight of each of 128 RGB colors was extracted and used as histogram bins. SIMPLIcity contains 1,000 images in 10 categories, and each category has 100 images. The images in this dataset are also represented as 128-bin histograms with the bins in three-dimensional space. We compared the performance of our method with three previous methods: lower-bounding based on dimensionality reduction (LB) [18], tree-based indexing (TBI) [8], and normal lower-bound indexing (NLI) [12]. For dimensionality-reduction in LB, we set the reduced histogram dimension n' that showed the best search performance in the experiments in [18]. We set n'= 18 for RETINA and n'=24 for Shader and SIMPLIcity, respectively. The same n' values were used in the proposed method. Table 1 shows the size of the M-tree indexes constructed in the proposed method.

Table 1. Size of the M-tree indexes for three datasets

	RETINA	SIMPLIcity	Shader
Data Size	1,391KB	412KB	12,154KB
Index Size	3,456KB	1,187KB	36,032KB

We used the precision, recall, and mean absolute percentage error (MAPE) as the accuracy measures. The precision and recall measures were used on the SIMPLIcity. As described above, the SIMPLIcity is composed of 10 categories, and each category contains 100 images. The category information is used as the ground truth to measure the precision and recall defined as follows [16]:

$$Precision = \left|\frac{\{data\ in\ each\ category\} \cap \{retrieved\ data\}}{\{retrieved\ data\}}\right|$$
$$,\quad Recall = \left|\frac{\{data\ in\ each\ category\} \cap \{retrieved\ data\}}{\{data\ in\ each\ category\}}\right| \tag{6}$$

The MAPE is used to evaluate the accuracy of the approximate method and defined as the following [16], [25]:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{a_i - f_i}{a_i}\right| \tag{7}$$

where a_i is the real EMD and f_i is the AEMD between two objects. In our experiments, the MAPE values are multiplied by 100 to show in the unit of percent. We used the total elapsed time as the performance measure. To obtain the more accurate results, we averaged the results on 100 random queries for each experiment. The experiments were performed on Microsoft Windows 7 on a workstation equipped with an Intel Core i5 2.80 GHz CPU and 4GB main memory.

5.2. Experimental Results

We evaluate the accuracy of the AEMD in the first and second experiments and then verify the superiority of the proposed method in the remaining experiments. We use only RETINA and Shader in evaluating the search performance, because SIMPLIcity has the histogram distribution similar to Shader and contains much smaller number of objects. As mentioned in Section 4.1, SIMPLIcity is used to measure the precision and recall of the proposed method. In the first experiment, we measured the MAPE of the AEMD, and the result is shown in Table 2.

Table 2. MAPE of AEMD

	RETINA	SIMPLIcity	Shader
AEMD	27.6%	21.4%	19.2%
AEMD (with MCW elimination)	14.4%	11.7%	9.8%

As shown in the table, the original AEMD incurs considerable errors for every dataset. Furthermore, the errors of the AEMD are larger than those reported in [16] because our datasets consist of higher dimensional histograms. In contrast, the errors of the AEMD are reduced nearly to half when the MCW elimination was employed. This is because the MCW elimination removes the histogram bins that may cause the errors in the AEMD computations. In all the experiments hereafter, we used the AEMD with the MCW elimination.

In the second experiment, we measured the ratio of object pairs satisfying the triangular inequality of the AEMD. As mention in Section 4.2, the AEMD does not generally satisfy the triangular inequality due to the errors between the AEMD and the EMD. We used Shader since its size is the largest. The experiment was performed with respect to a varying number of objects pairs, and its result is shown in Table 3.

Table 3. MAPE of AEMD

Number of comparisons	1,000	5,000	10,000	20,000	30,000
Satisfaction ratio	100%	99.9%	99.9%	99.8%	99.8%

As shown in Table 3, there exist only few cases violating the triangular inequality. This is because the error between the EMD and the AEMD for a pair (P, R) is not larger than the errors generated by the pairs (P, Q) plus (Q, R). In order to examine in a different viewpoint, we calculated the standard deviation of the MAPE, and the result is shown in Table 4. We can find in the table that the standard deviation of the errors is quite small. That is, the errors by the AEMD are mostly constant. Therefore, the false dismissals by the AEMD by violating the triangular inequality are not significant.

Table 4. Standard deviation of MAPE

	RETINA	SIMPLIcity	Shader
AEMD	0.0343	0.0320	0.0303
AEMD(with MCW elimination)	0.0235	0.0216	0.0211

In the third experiment, we measured the search performance and the accuracy of the proposed method while changing the parameter k'. The proposed method searches k' ($> k$) candidates based on the AEMD in the M-tree and then performs the post-processing based on the EMD to find k objects nearest from the query. In this experiment, we set k as 20 and the accuracy is defined as the ratio of common objects between the k-NN result based on the EMD and the search result of the proposed method. Table 5 shows the result. The result shows that both the accuracy and the elapsed time increase as k' increases. The accuracy is improved with the increase of k' because the false dismissals due to the AEMD are reduced. The execution time also increases slowly with the increase of k' due to the fast computation of the AEMD. In the experiments hereafter, we set $k' = 4k$ for RETINA and $k' = 2k$ for Shader.

Table 5. Accuracy and search performance of the proposed method with respect to varying *k'* values

(a) RETINA

	k'=k	*k'=2k*	*k'=3k*	*k'=4k*	*k'=5k*
Accuracy	88.3%	92.3%	94.7%	96.5%	97.3%
Elapsed time(seconds)	0.36	0.42	0.45	0.47	0.50

(b) Shader

	k'=k	*k'=2k*	*k'=3k*	*k'=4k*	*k'=5k*
Accuracy	95.4%	99.4%	99.8%	100%	100%
Elapsed time(seconds)	2.35	2.64	2.91	3.12	3.24

In the fourth experiment, we measured the precision and recall of the proposed method and compared with the search result based on the EMD using SIMPLIcity in order to verify the robustness of the proposed method. In this experiment, we set $k' = 2k$ since SIMPLIcity is very similar to Shader. As a result, in Table 6, for all k' values, the proposed method showed almost the same result as the EMD-based search. Such a good result of the proposed method is obtained by reducing the errors of the AEMD by the MCW elimination and the false dismissals by k'-NN search on the M-tree.

Table 6. Precision and recall of the proposed method and EMD-based search

(a) Precision

	k=10	*k=20*	*k=30*	*k=40*	*k=50*
Proposed method	88.2%	75.0%	66.1%	54.4%	44.5%
EMD-based search	88.2%	75.0%	66.1%	54.5%	44.8%

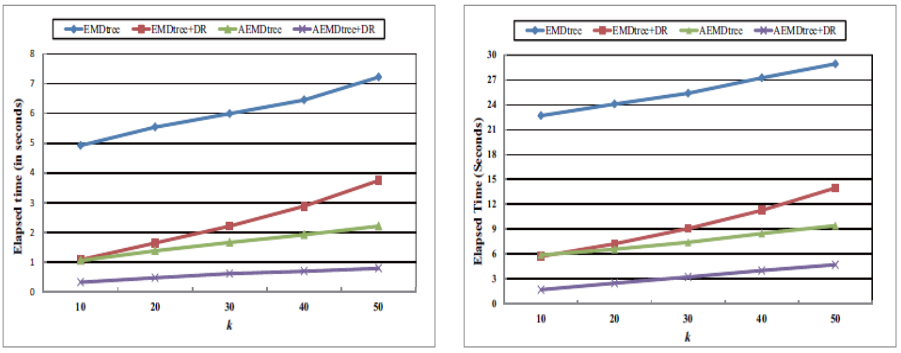
(b) Recall

	k=10	*k=20*	*k=30*	*k=40*	*k=50*
Proposed method	8.8%	15.0%	20.0%	27.2%	44.5%
EMD-based search	8.8%	15.0%	20.0%	27.3%	44.8%

As discussed in Section 4, the proposed method employs the three component techniques, namely the M-tree indexing, dimensionality-reduction, and the AEMD. In the fifth experiment, we examined the search performance due to various combinations of the three techniques in order to analyze the gain by each of them. The combinations in this experiment are summarized in Table 7.

Table 7. Combinations of component techniques

EMD_{tree}	Step 1: Build the M-tree with original histograms *H* Step 2: Perform *k*-NN_Search on the M-tree based on the EMD
$EMD_{tree+DR}$ (*k*-NN_EMD)	Step 1: Build the M-tree with dimensionality-reduced histograms *H'* Step 2: Perform *k*-NN_Search on the M-tree based on the EMD to find candidates Step 3: Refine *k*-NN by computing the EMD with the original histograms *H*
$AEMD_{tree}$	Step 1: Build the M-tree with the original histograms *H* Step 2: Perform *k'*-NN_Search on the M-tree based on the AEMD Step 3: Perform the post-processing based on the EMD on the original histograms *H*
$AEMD_{tree+DR}$ (our method)	Step 1: Build the M-tree with dimensionality-reduced histograms *H'* Step 2: Perform *k'*-NN_Search on the M-tree based on the AEMD to find candidates Step 3: Refine *k'*-NN by computing the AEMD with the original histograms *H* Step 4: Perform the post-processing based on the EMD on the original histograms *H*



(a) RETINA (b) Shader

Fig. 7. Comparison of search performance: the proposed AEMDtree+DR achieves the best performance

Figure 7 shows the results. Although the EMD_{tree} reduces the disk access overhead by using the M-tree, it shows the worst search performance because of the EMD computation overhead. The $EMD_{tree+DR}$, the *k*-NN_EMD in Figure 4, shows a better

performance than the EMD_{tree} because it reduces the EMD computation overhead in the M-tree search with the dimensionality-reduction. However, the overhead of computing the EMD is still very high in the post-processing step, since the number of candidates returned from the M-tree search is much larger than *k*. The $AEMD_{tree}$ performs the *k*-NN search on the M-tree based on the AEMD and thus achieves the faster search performance than the above two combinations. However, the $AEMD_{tree}$ suffers from the computational overhead due to the high-dimensional histograms in the M-tree search. On the other hand, the proposed method, the $AEMD_{tree+DR}$, reduces both the disk access overhead and the EMD computation overhead by combining the three component techniques and thus achieves the best search performance.

Table 8. Distribution of elapsed time of each step in three combinations (in seconds, on Shader)

(a) $EMD_{tree+DR}$

	M-tree search		*k*-NN refinement time	Total elapsed time
	Search time	I/O reads		
k=10	2.40	291.95	3.32	5.72
k=20	3.09	329.90	5.99	9.08
k=30	3.99	351.75	9.99	13.98

(b) $AEMD_{tree}$

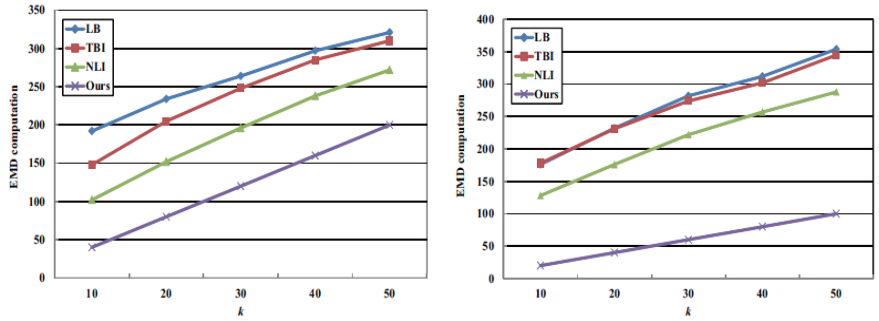
	M-tree search		Post-processing time	Total elapsed time
	Search time	I/O reads		
k=10	5.64	472.10	0.25	5.89
k=20	6.69	547.85	0.73	7.42
k=30	8.18	621.85	1.26	9.44

(c) $AEMD_{tree+DR}$

	M-tree search		*k'*-NN refinement time	Post-processing time	Total elapsed time
	Search time	I/O reads			
k=10	1.40	304.28	0.25	0.25	1.90
k=20	1.94	342.40	0.57	0.73	3.24
k=30	2.62	379.40	0.83	1.26	4.71

Table 8 shows the time (in the unit of seconds) elapsed in each step except the first M-tree building shown in Table 7 for three combinations $EMD_{tree+DR}$, $AEMD_{tree}$, and $AEMD_{tree+DR}$. The time is shown only for Shader, since RETINA shows very similar distribution of elapsed time.

As shown in the table, the $EMD_{tree+DR}$ completes the M-tree search quickly using the dimensionality-reduction. However, the total elapsed time is large due to the high EMD computation overhead in the refinement step. The $AEMD_{tree}$ searches for the k'-NN objects ($k' > k$) from the M-tree with the original high-dimensional histograms and thus has both the M-tree search time and I/O costs larger than the $EMD_{tree+DR}$. Nevertheless, the $AEMD_{tree}$ has the smaller elapsed time, since its post-processing needs to compute the EMD only for k' objects retrieved from the M-tree. The $AEMD_{tree+DR}$ has the smallest execution time even though it consists of one more k'-NN refinement step, which finds k'-NN objects from the candidates retrieved from the M-tree based on AEMD and thus is completed very quickly as shown in Table 8(c). It also needs to compute the EMD only for k' objects in the post-processing step, whose execution time is very short as in $AEMD_{tree}$.

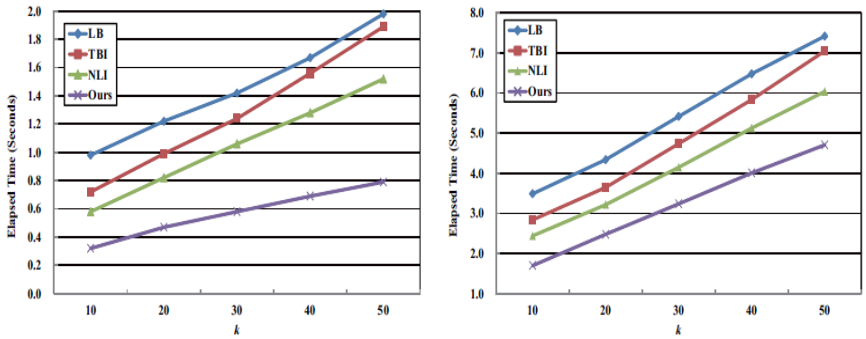


(a) RETINA (b) Shader

Fig. 8. Comparison of the number of EMD computations with respect to varying values of k: our method always has the smallest number of EMD computations

In the final experiment, we compared the number of EMD computations and the k-NN search time of the proposed method with the previous methods, namely LB [18], TBI [8], and NLI [12]. Figure 8 compares the number of EMD computations with respect to varying values of k. The proposed method shows the smallest number of EMD computations. Compared with LB, TBI, and NLI, our method reduces the EMD computations by up to 79.2%, 72.9%, and 60.7% on RETINA, and up to 88.7%, 88.6%, and 84.2% on Shader, respectively. The previous methods first search a set of candidates using the index or the lower-bounding function and then compute the EMD of each candidate to find the final k-NN. They perform many EMD computations, because the number of candidates is very large. On the contrary, the proposed method needs to compute the EMD only for k' candidates in the post-processing step. Therefore, the number of the EMD computations of our method is much smaller than the previous methods.

Figure 9 compares the total search time. Compared with LB, TBI, and NLI, our method reduces the search time by up to 67.3%, 55.5%, and 44.8% on RETINA and up to 51.7%, 40.1%, and 31.3% on Shader, respectively. LB shows the worst performance: it suffers from the scalability problem since it does not use an index and also needs many EMD computations as shown in Figure 7. TBI and the NLI show the better performance than LB, since they use the indexes. However, they require a considerable number of EMD computations as shown in Figure 7, which causes performance degradation. The proposed method reduces not only the disk access overhead using the index but also the number of EMD computations using the AEMD and thus has the better performance than the previous methods. We claim that our method should be recognized as more useful than the others especially for the CBIR on very large multimedia databases.



(a) RETINA (b) Shader

Fig 9. Comparison of search time with respect to varying values of k: our method always has the best performance than the others

6. Conclusion

In this paper, we proposed an approximate *k*-NN search method for the EMD-based CBIR. To perform the efficient *k*-NN search, both the disk access overhead and the EMD computational overhead should be reduced. The proposed method adopts the M-tree, a distance-based index structure, to reduce the disk access overhead. When building the M-tree, our method reduces the number of bins of the histograms using the dimensionality-reduction. After constructing the M-tree, it finds a small number of candidates from the disk by using the M-tree and performs the post-processing on them. The proposed method uses the approximate EMD in index retrieval and post-processing to reduce the computational overhead of the EMD. Also, we proposed the *k'*-NN search and the maximum common weight elimination to compensate the errors caused by the approximation. The extensive experiments reveal that our method achieves the significant improvement in terms of the number of the EMD computations and the *k*-

NN search time. The proposed method improves the performance of the previous method by up to 67.3% and only incurs 3.5% error. The retrieval results of the proposed method in real-world databases are almost the same as those of the original EMD, which indicates the errors of the proposed method hardly influence the quality of CBIR results. Given the fast processing time and small errors, our method can be used effectively in the CBIR for large databases.

Acknowledgements. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2017R1D1A1B03030969). This research was supported by the MSIT (Ministry of Science, ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2013-1-00881) supervised by the IITP (Institute for Information & communication Technology Promotion) and supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. NRF-2017M3C4A7083678).

A APPENDICES

Proof of Lemma 2:

Let the flow matrix $\widehat{\boldsymbol{F}} = [\hat{\boldsymbol{f}}_{\boldsymbol{ij}}]$ be the approximate minimum work of the AEMD. The AEMD can be defined as follows:

$$AEMD_D = \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{f}_{ij} d_{ij}$$

The $AEMD_{reduced}$ with the reduction matrix R is defined as follows:

$$AEMD_{D'} = \sum_{i'=1}^{n'}\sum_{j'=1}^{n'} \hat{f}_{i'j'}\, d'_{i'j'}$$

$$\hat{f}_{i'j'} = \sum_{\{i|r_{ii'}=1\}}\sum_{\{j|r_{jj'}=1\}} \hat{f}_{ij}$$

$$d'_{i'j'} = \min\{d_{ij} | r_{ii'} = 1 \wedge r_{jj'} = 1\}$$

Based on these equations, we can infer the following equations:

$$AEMD_D = \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{f}_{ij} d_{ij}$$

$$= \sum_{i'=1}^{n'}\sum_{j'=1}^{n'} \left(\sum_{\{i|r_{ii'}=1\}} \sum_{\{j|r_{jj'}=1\}} \hat{f}_{ij} d_{ij} \right) \tag{A.1}$$

$$\geq \sum_{i'=1}^{n'}\sum_{j'=1}^{n'} \left(\left(\sum_{\{i|r_{ii'}=1\}} \sum_{\{j|r_{jj'}=1\}} \hat{f}_{ij} \right) \times \min\{d_{ij} | r_{ii'} = 1 \wedge r_{jj'} = 1\} \right) \tag{A.2}$$

$$= \sum_{i'=1}^{n'}\sum_{j'=1}^{n'} \left(\left(\sum_{\{i|r_{ii'}=1\}} \sum_{\{j|r_{jj'}=1\}} \hat{f}_{ij} \right) \times d'_{i'j'} \right) \tag{A.3}$$

$$= \sum_{i'=1}^{n'}\sum_{j'=1}^{n'} \hat{f}_{i'j'} d'_{i'j'} = AEMD_{D'} \tag{A.4}$$

By using the reduction matrix, the $AEMD_D$ can be represented as Equation (A.1). If we replace d_{ij} by the minimum value that is satisfied $\{r_{ii'} = 1 \wedge r_{jj'} = 1\}$ as shown in Equation (A.2), the result is less than or equal to the $AEMD_D$. Since $d'_{i'j'}$ denotes the minimum value of the optimal reduced distance matrix D' (in Equation (A.3)), the $AEMD_{reduced}$ is always less than or equal to the $AEMD_D$ (in Equation (A.4)).

References

1. Y. Liu, G. Zhange, W. Ma.: A survey of content-based image retrieval with high-level semantics. Pattern Recognition, 40 (1): 262-282. (2007)
2. W. Zhou, H. Li, Q. Tian.: Recent Advance in Content-based Image Retrieval: A Literature Survey. eprint arXiv:1706.06064[cs.MM]. (2017)
3. M. Yasmin, S. Mohsin, M. Sharif.: Intelligent Image Retrieval Techniques: A Survey. Journal of applied research and technology, 12(1): 87-103. (2014)
4. A. Alzubi, A. Amira, N. Ramzan.: Semantic content-based image retrieval: A comprehensive study. Journal of Visual Communication and Image Representation, 32: 20–54. (2015)
5. E. Rashedi, H. Nezamabadi-pour, S. Saryazdi.: A simultaneous feature adaptation and feature selection method for content-based image retrieval systems. Knowledge-Based System, 39: 85-94. (2013)
6. E. Yildizer, A. Balci, T. Jarada, R. Alhajj.: Integrating wavelets with clustering and indexing for effective content-based image retrieval. Knowledge-Based Systems, 31: 55-66. (2012)
7. P. Ciaccia, M. Patella, P. Zezula.: M-tree: An efficient access method for similarity search in metric spaces. Proceedings of Very Large Data Bases, 426-435. (1997)

8. J. Xu, Z. Zhang, A. Tung, G. Yu.: Efficient and effective similarity search over probabilistic data based on earth mover's distance. Proceedings of International Conference on Very Large Data Bases,758-769. (2010)
9. I. Assent, A. Wenning, T. Seidl.: Approximate techniques for indexing the earth mover's distance in multimedia databases. Proceedings of IEEE International Conference on Data Engineering, 11. (2006)
10. H. Chen, et al.: A novel bankruptcy prediction model based on an adaptive fuzzy k-nearest neighbor method. Knowledge-Based Systems, 24 (8): 1348-1359. (2011)
11. T. Seidl, H. Kriegel.: Optimal multi-step k-nearest neighbor search. Proceedings of ACM International Conference on Management of Data, 154-165. (1998)
12. B. Ruttenberg, A. Singh.: Indexing the earth mover's distance using normal distributions, Proceeding of International Conference on Very Large Data Bases, 205-216. (2012)
13. I. Assent, M. Wichterich, T. Meisen, T. Seidl.: Efficient similarity search using the earth mover's distance for large multimedia databases. Proceedings of IEEE International Conference on Data Engineering, 307-316. (2008)
14. Y. Rubner, C. Tomasi, J. Guibas.: The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision, 40 (2): 99-121. (2000)
15. S. Shirdhonka, D. Jacobs.: Approximate earth mover's distance in linear time. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 1-8. (2008)
16. M. Jang, S. Kim, C. Faloutsos, S. Park.: A linear-time approximation of the earth mover's distance. Proceedings of ACM International Conference on Information Knowledge Management, 505-514. (2011)
17. V. Ljosa, A. Bhattacharya.: Indexing spatially sensitive distance measures using multi-resolution lower-bounds. Proceedings of EDBT International Conference on Extending Database Technology, 865-883. (2006).
18. M. Wichterich, I. Assent, P. Kranen, T. Seidl.: Efficient emd-based similarity search in multimedia databases via flexible dimensionality reduction. Proceedings of ACM International Conference on Management of Data, 199-211. (2008)
19. B. Moon, et al.: Analysis of the clustering properties of the Hilbert space-filling curve. IEEE Transactions on Knowledge and Data Engineering, 13 (1): 124-141. (2001)
20. C. Papadimitrous, K. Steiglitz.: Combinatorial Optimization: Algorithms and Complexity. 1st ed. NewYork: Dover Publications. (1998)
21. Y. Tang, et al.: The earth mover's distance based similarity search at scale. Proceedings of the VLDB Endowment, 313-324. (2013)
22. N. Beckmann, H. Kriegel, R. Schneider, B. Seeger.: The r*-tree: An efficient and robust access method for points and rectangles. Proceedings of ACM International Conference on Management of Data, 322-331. (1990)
23. M. Jang, et al.: On extracting perception-based features for effective similar shader retrieval. Proceedings of IEEE International Conference on Computer Software and Applications, 103-107. (2011)
24. J. Wang, J. Li, G. Wiederhold.: Simplicity: Semantics-sensitive integrated matching for picture libraries. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (9): 947-963. (2001)
25. J. Shepperd, C. Schofield.: Estimating software project effort using analogies. IEEE Transactions on Software Engineering, 23 (11): 736-743. (1997)
26. D. C. G. Pedronette, R. S. Torres.: A correlation graph approach for unsupervised manifold learning in image retrieval tasks. Neurocomputing Joural, 208: 66-79. (2016)
27. S. Sun, Y. Li, W. Zhou, Q. Tian, H. Li.: Local residual similarity for image re-ranking. Information sciences, 471: 143-153. (2017)
28. J. Wu, Y. He, X. Qin, N. Zhao, Y. Sang.: Click-Boosted Graph Ranking for Image Retrieval. Computer Science and Information Systems, Vol. 14, No. 3, 629–641. (2017)

Min-Hee Jang earned the M.S. and Ph.D. degrees in electronics and computer engineering from Hanyang University, Korea, at 2006 and 2012, respectively. In 2013, he joined Carnegie Mellon University, Pittsburgh, USA, at the Computer Science Department as a postdoctoral researcher. He is currently working at Samsung Electronics as a software engineer and data analyst from the summer of 2014. His research interests include data mining, multimedia information retrieval, and social network analysis.

Sang-Wook Kim received the BS degree in Computer Engineering from Seoul National University, Seoul, Korea in 1989 and earned the MS and PhD degrees from Korea Advanced Science and Technology (KAIST) at 1991 and 1994, respectively. He is Professor at Department of Computer Science and Engineering, Hanyang University, Seoul, Korea. Professor Kim worked with Carnegie Mellon University and IBM Watson Research Center as a visiting scholar. He is an associate editor of Information Sciences. His research interests include data mining and databases.

Woong-Kee Loh received his B.S. (1991), M.S. (1993), and Ph.D. (2001) degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), South Korea. Currently, he is an associate professor in Dept. of Software, Gachon University, South Korea. He has served as a program and an organizing committee member in many international conferences including DaWaK 2007-08, CIKM 2009, PAKDD 2011-14, ICDE 2015, SSTD 2015, DASFAA 2017, and BigComp 2017-19. His research interests include massively parallel large-scale data mining.

Jung-Im Won received her M.S. and Ph.D. degrees in computer engineering from the Hallym University, Korea, in 1997 and 2004, respectively. Now she is a research professor in Hallym University in Korea. Her research interest includes database system, data mining and bioinformatics.

Received: October 10, 2018; Accepted: March 25, 2019