# MSIDX: Multi-Sort Indexing for Efficient Content-based Image Search and Retrieval

Eleftherios Tiakas, Dimitrios Rafailidis, Anastasios Dimou and Petros Daras, *Member, IEEE*

***Abstract*—In this paper, a novel approximate indexing scheme for efficient content-based image search and retrieval is presented, called Multi-Sort Indexing (MSIDX). The proposed scheme analyzes high dimensional image descriptor vectors, by employing the value cardinalities of their dimensions. The dimensions' value cardinalities, an inherent characteristic of descriptor vectors, are the number of discrete values in the dimensions. As expected, value cardinalities significantly vary, due to the existence of several extraction methods. Moreover, different quantization and normalization techniques used in the extraction process, have a strong impact on the dimensions' value cardinalities. Since dimensions with high value cardinalities have more discriminative power, a multiple sort algorithm is used to reorder the descriptors' dimensions according to their value cardinalities, in order to increase the probability of two similar images to lie within a close constant range. The expected bounds of the constant range are defined in detail, following deterministic and probabilistic analyses. The proposed scheme is fully suitable (a) for real-time indexing of images, and (b) for searching and retrieving relevant images with an efficient query processing algorithm. In our experiments with five real datasets, we show the superiority of the proposed approach against hashing methods, also suitable for approximate similarity search.**

***Index Terms*—Multi-sort, indexing, content-based image retrieval, approximate similarity search.**

## I. Introduction

THE availability of images over the Internet emerged the imperative need to address the challenge of content-based searching, in order to find visually similar content. Exhaustive search is infeasible for large scale applications due to its extensive time requirements. Thus, indexing methods are needed, able to provide efficient search time and retrieval accuracy. However, multimedia objects like compressed images, video and audio streams are usually described by sequences of descriptor vectors with over than a thousand dimensions, and their similarity is examined by nearest neighbor search. In this high dimensional space, the performance of existing widely studied tree-based indexing methods deteriorate significantly, due to the well-known problem of Dimensionality Curse [29], [10].

In the last decade, many dimensionality reduction methods [23], [27], [59], [28], [51], [2], [46], [4] and hashing techniques [19], [15], [13], [45], [43], [38], [32], [57], [34], [44], [30], [50], [58], [42], [31], [36], [56], [24], [33], [25] have been proposed to overcome the problems of the tree-based methods and thus to provide efficient solutions for high-dimensional data. In the first case, dimensionality reduction methods try to reduce the number of dimensions of the high-dimensional data. In particular, the data are transformed into a lower-dimensional space by using dimensionality reduction methods and then an index is built on it. In the second case, the encoding of such data into binary codes using appropriate hash functions enables higher scalability due to the compactness of the data and their efficient indexing. Similar high-dimensional objects are mapped to similar binary codes. Therefore, approximate nearest neighbor search is performed by only examining similar binary codes. However, the hashing methods often fail to achieve high accuracy in their approximations, especially when the hashing functions are drawn independently from the data, or when a short binary code length is selected. Moreover, for long binary code lengths a significant preprocessing time is required.

In the proposed MSIDX method, a new key factor is exploited, which to the best of our knowledge, has not been explored so far. In particular, the motivation for designing a novel indexing scheme is the correlation between the value cardinality in every dimension of the descriptor and the discriminative power of the specific dimension, assuming that dimensions with high value cardinalities have more discriminative power. This correlation is explored by examining the content-based descriptor vector extraction process of images.

### A. Analysis of the Dimensions' Value Cardinalities of Descriptor Vectors

The value cardinalities of the descriptor in each dimension is the number of unique discrete values that can be found in the examined dimension throughout a database of descriptor vectors. The range and the density of these observed values is strongly affected by the extraction strategy followed for each descriptor type. In this section, the impact of the descriptor extraction techniques on the dimensions' value cardinalities is further analyzed and discussed.

Descriptor vectors of images, representing either a local area or the whole image, are extracted by forming histograms that describe the value distribution of a specific attribute, which defines the unique characteristics of the descriptor. Due to differences in scale, resolution, color distribution and illumination, the descriptor assigned to the same object or texture can vary significantly among different content. Moreover, in the case of creating a global image descriptor from local ones using a histogram-based technique, the population of the histogram is defined by the number of local descriptors. This number is affected by various design choices, such as the size of the image, the localization strategy (dense grid or interest points) and the extraction density in the selected strategy.

E. Tiakas, D. Rafailidis, A. Dimou and P. Daras are with the Information Technologies Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece. e-mail {tiakas,drafail,dimou,daras}@iti.gr

Manuscript received; revised

In order to produce comparable histograms for search and retrieval purposes, normalization [37] and descriptor value quantization techniques are applied during the post-processing phase. Using normalized vectors implies the use of the contour of the histogram, which is more robust to the above mentioned factors, rather than the absolute values of the histogram. It has been shown that applying normalization leads to higher accuracy in retrieval methodologies [49], [60]. After the normalization process, the descriptor values become real numbers. Unfortunately, floating-point arithmetics increase the complexity of the calculations, leading to increased processing time and storage needs, especially for large databases of images. Descriptor value quantization decreases the value cardinalities of the representation space through a destructive process, introducing a trade-off between accuracy and performance.

After applying this post-processing, the observed value cardinalities in each dimension of the descriptor change significantly. The design choice in the quantization strategy followed, is to define the theoretical bound to the cardinalities of the descriptor. In practice, though, this bound is reached for a very limited number of dimensions. In most dimensions the descriptor vectors values hold zero or are highly repetitive, restricted and non-distinctive. Therefore, these dimensions do not posses significant descriptive power and do not enhance the retrieval accuracy of the system. This observation holds even more for descriptors with high dimensionality.

### B. *Contribution and Layout*

In this paper, a new indexing method is proposed, called MSIDX, which employs the value cardinalities of the descriptors' dimensions, achieving high retrieval accuracy and low search time. In particular, by reordering the dimensions of the descriptor vectors according to their value cardinalities based on the proposed multiple sort algorithm, MSIDX increases the probability of two similar images to lie within a close constant range. The expected bounds of the constant range are defined, following both deterministic and probabilistic analyses. MSIDX fully supports the following desired characteristics: (a) dynamic indexing and storing of the new image content, by supporting a real-time insertion technique, and (b) efficient search time and accurate retrieval. Through extensive experimental evaluation with five publicly available datasets, we demonstrate that the proposed MSIDX method significantly outperforms state-of-the-art hashing methods, also suitable for approximate similarity search.

The rest of the paper is organized as follows: after summarizing the related work in Section II, the proposed MSIDX scheme is described in Section III, while the mathematical formulation for determining the bound of the constant range is provided in Section IV. Experimental results are presented in Section V and finally, conclusions are drawn in Section VI.

## II. Related Work

### A. *Tree-based Methods*

Several tree-based indexing methods have been proposed for the problem of nearest neighbor search, such as: KD-trees [18], [5], [6], R-trees [22], M-Trees [14], Quad-Trees [47], Vantage Point Trees (VPT) [12], Voronoi Trees (VT) [16], etc. Additionally, several tree-based indexing methods for approximate nearest neighbor search have also been proposed, such as: Spatial Approximation Tree (SAT) [40], Approximating Eliminating Search Algorithm (AESA) [55]. Extensive surveys for the most tree-based methods can be found in [48], [11], [8].

The main strategy for all tree-based indexing methods is to prune tree branches on the established bounding distances in order to reduce the node accesses. However, in high-dimensional spaces, where the multimedia objects lie, the tree-based indexing methods are inefficient, performing worse than exhaustive search [7].

### B. *Dimensionality Reduction Methods*

Instead of indexing the data into the original high-dimensional space, dimensionality reduction methods aim at mapping the data into a lower-dimensional subspace. The main idea is to make such a transformation without losing much information and build an index on the subspace. Many local and global dimensionality reduction methods have been proposed, such as the works in [23], [27], [59], [28], [51], [2], [46]. Global dimensionality reduction methods map the whole dataset into a much-lower dimensional subspace. For example, the Isometric Feature Mapping method [2] estimates geodesic distances and uses them to project the data into the embedded space. Local dimensionality reduction methods divide the dataset into correlated clusters and then each cluster is reduced in subspaces independently. For example, the Locally Linear Embedding method [46] projects the data to a low-dimensional space, while preserving local geometric properties.

The preprocessing cost of such transformations is often high, due to dense matrix operations (especially products, eigenvector and eigenvalue calculations). Several optimization techniques have been proposed to reduce the preprocessing cost and keep the accuracy of the estimated distances [28], [51]. For example, the Laplacian Eigenmaps method [4] uses an extra weighted distance between the data points as a loss function in order to evaluate and optimize the dimensionality reduction results.

Dimensionality reduction methods can be used either for approximate or exact similarity search. In the first case, the similarity search is performed only into the transformed subspace. In the second case, firstly the similarity search is performed into the transformed space, where lower bounds on the distances are used for filtering, then a resulting set of candidates is returned, and finally the candidates are refined in the original space with exact search (e.g. [28]).

In the case of exact similarity search, the recently proposed Data Co-Reduction (DCR) method of [28], significantly outperforms the existing methods for lossless retrieval, especially in the presence of extremely high dimensionality. The DCR method simultaneously reduces both the size and the dimensionality of the original data into a compact subspace, where lower bounds of the actual distances in the original space can be efficiently established to achieve fast and lossless similarity search in a filter-and-refine approach.

In the case of approximate similarity search, it has been demonstrated that the dimensionality reduction methods return accurate results in relative low dimensional spaces [51]. However, MSIDX also offers an approximate solution that significantly reduces the search time, while the computational cost of additional data transformations of the dimensionality reduction methods is avoided.

### *C. Hashing Methods*

The hashing methods have proven to be suitable for approximate similarity search, since they support efficient indexing and data compression. The basic idea of the hashing methods is (a) to encode the distances between the data into the form of compressed sequences of bits by using hash functions, and (b) to store the encoding distances into buckets, in order to ensure that the probability of collision is much higher for data that are close to each other than those that are far apart. Then, they approximate exact similarity measures by comparing hash codes, using a hamming distance on binary codes or other measures. Different strategies are followed during the preprocessing for the generation of the binary codes. The existing hashing methods can be broadly categorized as data-independent and data-dependent.

In data-independent hashing methods, the hashing functions are defined independently from the data. One of the most popular method is Locality Sensitive Hashing (LSH) [19], which is based on projection onto random vectors drawn from a specific distribution. Many hashing methods, where also a randomized process is followed in various metric spaces, are: p-stable LSH [15], min-hash [13], Shift-Invariant Kernels Hashing [45], Entropy based LSH [43], Multi-Probe LSH [38], posteriori Multi-Probe LSH [32], etc. Such methods are based on random projections and according to the Johnson-Lindenstrauss Theorem [35], at least $O(\ln n/\epsilon^2)$ projection vectors are required (where $n$ is the dataset size), so as to preserve the pairwise distances with a relative error $\epsilon$. Therefore, in order to decrease the relative error and increase the probability that similar objects have similar hash codes, the random projection based methods require many random vectors to generate the hash tables (each table corresponds to one random vector), leading to a large storage space and a high computational cost. Alternatively, several methods instead of following randomization strategies use deterministic structuring based methods like grids [57], space filling curves [34], [44], or lattices [30], [50]. Since the efficiency of the deterministic structuring methods depend on the data distributions, the randomized processes are more adaptive and usually more efficient. However, the data-independent hashing methods are often inefficient, especially for short lengths of binary codes, due to the fact that their hashing functions are drawn independently from the data.

In data-dependent hashing methods, the hashing functions are defined only for a preselected training dataset, which is usually a subset of the data, and involve similarity calculations for the training dataset. They try to fit the data distribution to the feature space in order to group the similar items and preserve locality. Notable examples of data-dependent hashing methods are: Spectral Hashing [58], which is based on spectral graph partitioning; K-means based hashing [42], which uses K-means clustering in the generation process of the binary codes; Subspaces Product Quantization [31], which decomposes the feature space into a Cartesian product of low-dimensional subspaces, each subspace is quantized separately, and the asymmetric distances are computed between the query and the quantized codes with the help of lookup tables; Kernelized Locality-Sensitive Hashing [36], which generalizes hashing to any Mercer kernel; Semi-Supervised Hashing [56], which exploits label information of the training set; Multiple Feature Hashing [53], which combines multiple features (i.e., global feature HSV color histogram and local visual features LBP) of videos, in order to learn the hash codes of the training keyframes and a series of hash functions in a joint framework; Iterative Quantization [20], which minimizes quantization error by rotating zero-centered PCA projected data; Joint Optimization [24], which jointly optimizes both search accuracy and search time using compact binary codes; Random Maximum Margin Hashing [33], which constructs hash functions by using large margin classifiers with arbitrarily sampled data points that are randomly separated into two sets.

Efficiency improvements of data-dependent methods over independent ones have been shown in several studies [31], [58], on condition that limited hash code sizes are considered. This happens because by increasing the number of hash functions there is a lack of independence between them. For example, Spectral Hashing [58] outperforms many data-independent methods for small code sizes, but it is outperformed by the data-independent method of Shift-Invariant Kernel Hashing [45] for sizes over 64 bits. Moreover, in all data-dependent hashing methods there is often a significant preprocessing cost for learning the selected training dataset and for generating the binary codes.

In all aforementioned hashing methods, the most common technique for assigning the binary codes is to partition the metric space of the projected data points with appropriate hyperplanes and set two different codes for each side. In a very recent approach, Spherical Hashing [25], hypershperes (instead of hyperplanes) are used, so as to partition the data points and to compute the binary codes. In the experimental evaluation of [25], authors showed that Spherical Hashing outperforms other state-of-the-art hashing methods.

## III. The Proposed MSIDX Method

The key idea of the proposed MSIDX method is: *to reorder the storage positions of images' descriptors according to value cardinalities of their dimensions, by performing a multiple sort algorithm, in order to increase the probability of having two similar images in storage positions that do not differ more than a specific global constant range, denoted by a parameter $W$.*

In particular, a dataset of images' descriptor vectors, which are stored either in the memory or in the disk, are reorganized based on the value cardinalities of the dimensions. Thus, each new image is inserted and allocated in a specific storage position in the existing dataset of descriptor vectors. The

MSIDX approximate indexing scheme is designed to support the following functionalities: (a) **preprocessing** for indexing and storing of a pre-existing image dataset in the form of high dimensional descriptor vectors, (b) **insertion** for real-time indexing and storing of a new (non-existing) image in the form of descriptor vector, (c) **query processing** for searching of the top-$k$ similar images to a posed image query and (d) **deletion** for removing already indexed images. In the rest of this section, the respective algorithms are described, accompanied with their implementation details and complexity analysis, followed by the space requirements of the MSIDX indexing scheme.

## *A. Preprocessing Algorithm*

**Algorithm Description**

Let $N$ be the total number of the available images $o_j$, $j = 1, ..., N$, from which a set $\mathbf{S}$ of $D$-dimensional descriptor vectors is extracted with values $v_{ij}$, $i = 1, ..., D$. From now on when we are referring to an image $o_j$ we imply its descriptor vector, thus $o_j \in \mathbf{S}$. After loading the dataset $\mathbf{S}$ in memory, each $D$-dimensional vector is stored in a specific storage position $pos_j$, which is equal to a pointer, linked to a physical memory address. According to the position $pos_j$, each image $o_j$ is connected with the previous $o_{j-1}$ and the next stored image $o_{j+1}$ in a doubly linked list $\mathbf{L}$.

The outline of the MSIDX preprocessing algorithm is depicted in Figure 1. In line 1, the value cardinality $c_i$ is computed for each dimension $i$, where three separate cases can be identified depending on the descriptor value type.

(a) **Integer values**: In case that the values are integer, only the different values are taken into account, and the total count is the value cardinality $c_i$.
(b) **Normalized real values**: In case that the values are real, produced by value normalization techniques of previously integer values as described in Section I-A, the calculation strategy of the value cardinality $c_i$ is the same with the case of integer values, due to the restricted value cardinality of the original integer-valued descriptor vector.
(c) **Real values**: In case that the extraction process of the descriptor generates real values, the calculation strategy of the value cardinality $c_i$ is performed after limiting the decimal accuracy of the descriptor values, as in the case of integer values. However, in practice, the extracted descriptors have a limited decimal accuracy, usually between 4 and 6 decimals, due to space and computational restrictions. It is stressed that in our experiments no additional value quantization was used.

In line 2, the priority index $p_i$ is created, based on the value cardinalities $c_i$. Since dimensions with high value cardinalities correspond to dimensions with high discriminative power, the value cardinalities $c_i$ are sorted in a descending order. The priority index $p_i$ is the position of the value cardinality $c_i$ in the sorted value cardinalities. Thus, high value cardinality $c_i$ of the dimension $i$ corresponds to high priority index $p_i$. Then, in line 3, except for the strong ordering criterion of value cardinalities $c_i$, an additional ordering criterion $ex$ could be used, stored into the extra dimension $D + 1$ of image's $o_j$ descriptor vector. Therefore, in line 4, a specific priority index $p_k$ (with $k = D + 1$) is computed for the extra dimension $D + 1$. It is important to note that the steps in lines 3 and 4 are optional and the calculation of the values $ex$ in the extra dimension $D + 1$ and its respective priority index $p_k$ are completely free to be selected according to the requirements of different domains and applications. For example, an aggregation of the descriptor vector values could be used in the optional dimension $D + 1$ to enhance the similarity/distance between the images. Consequently, in case of defining a high priority index $p_k$ for the optional dimension $D+1$, the images will be reordered based primarily on this aggregation. In line 5, the multiple sort algorithm is performed in the storage positions of images $pos_j$ according to the calculated dimensions priority indices $p_i$. In line 6, the reordered dataset $\mathbf{S}'$ is returned, while the images' $pos_j$ positions and their interconnections ($pos_{j-1}$, $pos_{j+1}$) are updated in the doubly linked list $\mathbf{L}$.

```
Algorithm Preprocessing-MSIDX(S)
Input: S = the set of $N$ $D$-dimensional descriptor vectors
Output: S' = the multiple-sorted set

01a. compute value cardinalities $c_i$, of the dimensions
01b.        $i = 1, ..., D$ of dataset S.
02a. sort the value cardinalities $c_i$ descending,
02b.     create a priority index $p_i$ on dimensions, $i = 1, ..., D$,
02c.        based on the sorted value cardinalities.
03a. [optional] calculate $ex$, for each image $o_j \in$ S,
03b.      $j = 1, ..., N$, store it in an extra dimension $D + 1$.
04a. [optional] define a specific priority index $p_k$,
04b.        with $k = D + 1$ for the $ex$ values.
05a. multiple sort the $N$ images, using the $p_i$ order
05b.        in all dimensions.
06a. update the positions in L and
06b.        return the multiple-sorted dataset S'.
```

Fig. 1. MSIDX preprocessing algorithm.

**Implementation Details and Complexity Analysis**

For each dimension, the value cardinality and the respective priority index are calculated in $O(N \log N)$, using a common set structure (binary tree, heap, etc.). Therefore, a complexity $O(D \cdot N \log N)$ is required, to compute the value cardinality and the priority index for all dimensions. The multiple sort procedure is performed by the quick sort algorithm, based on the comparative function of Figure 2, with complexity $O(D \cdot N \log N)$. Summarizing, the total complexity of the MSIDX preprocessing algorithm is $O(D \cdot N \log N)$.

## *B. Insertion Algorithm*

**Algorithm Description**

Since $\mathbf{S}'$ is reorganized based on the preprocessing algorithm, the MSIDX scheme supports insertions of new images, according to the algorithm depicted in Figure 3. In line 1, the optional value $ex$ of the new image $o_{new}$ is calculated and in

```
Function Compare-MSIDX(image $o_a$, image $o_b$)
Input: $o_a, o_b$: $D$-dimensional descriptor vectors
Output: 1 (if $o_a > o_b$), -1 (if $o_a < o_b$), 0 (if $o_a = o_b$)

01. $v_i$ = value of $o_a$ in dimension $i = 1, ..., D$.
02. $y_i$ = value of $o_b$ in dimension $i = 1, ..., D$.
03. for $i$ = 1 to D
04.     if $v_i > y_i$ then return 1
05.     if $v_i < y_i$ then return -1
06. end-for
07. return 0
```

Fig. 2. The compare function for the multiple sort.

line 2 the storage position $pos_{new}$ is allocated for inserting the new image $o_{new}$ to $\mathbf{S}'$, by using the compare function of Figure 2. In line 3, the new image $o_{new}$ is inserted into the allocated position $pos_{new}$, while the respective interconnection ($pos_{new-1}$, $pos_{new+1}$) is updated in the doubly linked list $\mathbf{L}$, and in line 4, the updated dataset $\mathbf{S}''$ is returned.

```
Algorithm Insert-MSIDX(image $o_{new}$)
Input: $o_{new}$ the new image in the form
of $D$-dimensional descriptor vector
Output: the updated dataset $\mathbf{S}''$

01a. [optional] compute the value $ex$ of the optional extra
01b.     dimension for image $o_{new}$.
02a. allocate the storage position $pos_{new}$ using the same
02b.     compare function in Figure 2.
03a. insert the image $o_{new}$ into the allocated position $pos_{new}$
03b.     and update the positions in L
04. return the updated dataset $\mathbf{S}''$.
```

Fig. 3. MSIDX insertion algorithm.

**Implementation Details and Complexity Analysis**

The optional value $ex$ of the new image $o_{new}$ is computed instantly from its descriptor vector values in $O(1)$. Therefore, the complexity of the insertion algorithm solely depends on the allocation method for detecting the storage position $pos_{new}$ for inserting the new image $o_{new}$ to the organized dataset $\mathbf{S}'$. There are two possible options for implementing the allocation method: (a) using an insertion-sort method with $O(N)$ complexity, which is not suitable for a large scale $\mathbf{S}'$ or (b) running the multiple quick sort algorithm in $\mathbf{S}'$, by adding the new image $o_{new}$ to the last storage position of $\mathbf{S}'$. The latter method outperforms the former, since $\mathbf{S}'$ is already multiple sorted based on the value cardinalities $c_i$ of the dimensions and thus, few swaps are required to allocate the new image's final storage position $pos_{new}$. In particular, the complexity of the insertion algorithm depends on the distribution type that the values $v_{ij}$ (with $i = 1, ..., D$ and $j = 1, ..., N$) follow within the dimensions. If the values $v_{ij}$ follow a near uniform distribution, then an average complexity of $O(D \cdot \log N)$ is required for the insertion algorithm. On the contrary, if the values $v_{ij}$ follow skewed distributions, then the worst case complexity can reach $O(D \cdot N \log N)$. To avoid the latter, a binary search approach over the values $v_{ij}$ is performed, before the insertion algorithm starts, by searching in each dimension $i$ according to the priority index $p_i$. Therefore, the insertion position $pos_{new}$ of the new image is allocated in $O(D \cdot \log N)$ whether the values $v_{ij}$ follow a uniform or a skewed distribution, resulting in real-time complexity of the insertion algorithm.

## C. Query Processing Algorithm

**Algorithm Description**

The query processing algorithm of MSIDX is presented in Figure 4. It is divided into the following parts: (a) in lines 1-3, the query image $o_q$ is inserted into the dataset $\mathbf{S}$ and its storage position $pos_q$ is allocated; (b) in lines 4-15, the distances between the images that lie within $W$ next and $W$ previous storage positions from the allocated storage position $pos_q$ are calculated and consequently, the IDs of these images denoted by $t_n$ (with $n = 1, ..., 2W$) and their corresponding distances from $o_q$ are inserted into a minimum-heap structure $\mathbf{H}$; (c) in lines 16-20, the top-$k$ images are retrieved and removed from the top of the heap $\mathbf{H}$, constituting the results set $\mathbf{R}$.

More specifically, line 1 checks if the query image $o_q$ is already stored in the dataset $\mathbf{S}$ and in line 3, the storage position $pos_q$ is allocated. Otherwise, the insertion algorithm of Figure 3 starts to index the query image $o_q$ to the dataset $\mathbf{S}$ and the respective storage position $pos_q$ is allocated. In line 2, the set $\mathbf{R}$ and the minimum-heap $\mathbf{H}$ are initialized. The set $\mathbf{R}$ returns the IDs of the top-$k$ similar images to the posed query image $o_q$, while the minimum-heap $\mathbf{H}$ is a structure prioritized by the computed distances $d$ between the inserted images $t_n$ in $\mathbf{H}$ and the query image $o_q$. Therefore, the image $t_1$ located on the top of the heap $\mathbf{H}$, has always the minimum distance from $o_q$. In Line 4, the main loop starts, where the $2W$ closest images' positions are retrieved from the reorganized dataset $\mathbf{S}$, constrained to $2W > k$. In lines 5-6, each image $o_a$ that lies in a relative position $pos_q - j > 0, j = 1, ..., W$ is detected and in line 7 its distance from the query image $o_q$ is calculated, based on the predefined distance measure $d$ (e.g. L1, L2, squared-L1, etc.). In line 8, the ID of the detected image $o_a$ and the corresponding computed distance $d(o_q, o_a)$ are inserted into the heap $\mathbf{H}$. Respectively, in lines 10-11, each image $o_b$ that lies in a relative position $pos_q + j \leq N, j = 1, ..., W$ is detected and in line 12 the distance from the query image $o_q$ is calculated. In line 13, the ID of the detected image $o_b$ and the corresponding computed distance $d(o_q, o_b)$ are inserted into the heap $\mathbf{H}$. Finally, in line 16-20 the top-$k$ images $t_n$ are retrieved and removed from the top of the heap $\mathbf{H}$, constituting the results set $\mathbf{R}$.

**Implementation Details and Complexity Analysis**

The complexity of the query processing algorithm is calculated as the aggregation of its constituting parts (a) allocating the storage position $pos_q$ (lines 1-3), (b) building the minimum-heap structure $\mathbf{H}$ (lines 4-15) and (c) generating the

Algorithm **MSIDX Query Processing Algorithm**($o_q$, $k$, $W$)
Input: $o_q$ the query image in the form
of a $D$-dimensional descriptor vector,
$k$ the number of top results,
$W$ the search radius
Output: the top-$k$ results set $\mathbf{R}$

```
01a. if the image $o_q$ does not exist in the dataset S then
01b.     run the insertion algorithm in Figure 3.
02. set $\mathbf{R} = \emptyset$ , min-heap $\mathbf{H} = \emptyset$ .
03. allocate the position $pos_q$ of the query image $o_q$ in S.
04. for $j$ = 1 to $W$
05.     if $pos_q - j > 0$ then
06.         $o_a$ = image in position $pos_q - j$.
07a.         compute the distance $d(o_q, o_a)$
07b.             using a predefined distance measure $d$.
08.         insert the ID of $o_a$ into H with distance $d(o_q, o_a)$.
09.     end-if
10.     if $pos_q + j \leq N$ then
11.         $o_b$ = image in position $pos_q + j$.
12a.         compute the distance $d(o_q, o_b)$
12b.             using the distance measure $d$.
13.         insert the ID of $o_b$ into H with distance $d(o_q, o_b)$.
14.     end-if
15. end-for
16. for $n$ = 1 to $k$
17.     retrieve and remove the image $t_n$ located in top of H.
18.     R = R $\cup \{t_n\}$.
19. end-for
20. return the top-$k$ results set R.
```

Fig. 4. MSIDX query processing algorithm.

result set $\mathbf{R}$ (lines 16-20). More precisely, in lines 1-3, the complexity of allocating the position $pos_q$ is $O(D \cdot \log N)$, based on the complexity analysis of the insertion algorithm in Section III-B.

In lines 4-15, the $2W$ closest images' positions are sequentially retrieved, which are located below and above the query image positions $pos_q$ in $O(2W \cdot D)$. Additionally, since $2W$ image IDs are inserted into the heap $\mathbf{H}$, each image's $t_n$ insertion into the heap is performed in $O(\log 2W)$. However, the complexity can be improved by maintaining into the heap only the $k$ most similar images to the query $o_q$, while varying the parameter $W$. In particular, let $o_c$ be the currently candidate image for being inserted into the heap $\mathbf{H}$, while $t_k$ is the image with the $k$ largest distance from the query $o_q$, which is currently stored into the heap $\mathbf{H}$. If the condition $d(o_q, o_c) < d(o_q, t_k)$ does hold, then $o_c$ is inserted into the heap and $t_k$ is removed, otherwise $o_c$ is discarded. Therefore, $k$ images are maintained into the heap $\mathbf{H}$ and thus, the complexity of each image $t_n$ insertion into the heap is reduced from $O(\log 2W)$ to $O(\log k)$. As a result, the total complexity of the main loop (lines 4-15) is $O(2W \cdot D \cdot \log k)$.

In lines 16-20, $k$ images are retrieved and removed from the top of the heap in $O(k)$. Summarizing, the final complexity of the MSIDX query processing for retrieving the top-$k$ similar images to a posed query image $o_q$ is:

$$O(D \cdot \log N) + O(2W \cdot D \cdot \log k) + O(k)$$

Thus, the value of the parameter $W$ plays an important role in the total complexity of the MSIDX query processing algorithm. For this reason, deterministic and probabilistic analyses are presented in Section IV to set the expected bounds of $W$, in order to ensure high retrieval accuracy and low complexity of the MSIDX query processing algorithm.

### *D. Deletion Algorithm*

According to the preprocessing and the insertion algorithms, each image $o_j$ is stored into a specific position $pos_j$ in the form of a pointer, linked to a physical memory address. The complexity of allocating the position $pos_j$ is $O(D \cdot \log N)$, based on the complexity analysis of the insertion algorithm in Section III-B. Thus, the deletion of $o_j$ is performed in $O(D \cdot \log N)$, by retrieving the physical memory address $pos_j$ and clearing its content.

### *E. Space Requirements*

In order to support the aforementioned functionalities, the MSIDX framework has the following space requirements: (a) $O(D \cdot N)$ space for loading the dataset $\mathbf{S}$, since it consists of $N$ images in the form of $D$-dimensional descriptor vectors, (b) $O(N)$ space for storing the $N$ positions $pos_j$ in the form of address pointers, (c) $O((D + 1) \cdot N)$ space (in case of calculating the optional value $ex$ or $O(D \cdot N)$, otherwise) for computing the value cardinalities in the preprocessing algorithm and (d) $O(k)$ space for the query processing algorithm, since a temporary space for constructing the heap structure $\mathbf{H}$ is required, which stores the IDs and the computed distances of the top-$k$ retrieved images, while varying the parameter $W$. Summarizing, the total space requirements are

$$O(D \cdot N) + O(N) + O((D + 1) \cdot N) + O(k)$$

However, in large scale datasets it does hold that $N \gg k$ and therefore, the total space requirements are $O(D \cdot N)$, equal to the space requirements for storing the dataset $\mathbf{S}$.

## IV. Mathematical formulation of MSIDX

In this section a deterministic and probabilistic analysis is presented to set the expected bounds of the $W$ parameter.

### *A. Deterministic Analysis*

Let **S'** be the reordered $D$-dimensional dataset of $N$ images $o_j$, with $j = 1, ..., N$, produced by the preprocessing and the insertion algorithms. As already described in Section III-A, the cardinalities have been sorted in a descending order, since dimensions with high value cardinalities correspond to dimensions with high discriminative power. Thus, let $c'_i$, with $i = 1, ..., D$, be the sorted value cardinalities where: $c'_1 \geq c'_2 \geq ... \geq c'_D$.

Let $v_{ij}$, with $j = 1, ..., c'_i$, be the descriptor vector's values stored in each dimension. Let also $g$ be the cardinality of the equal-value groups, produced by performing multiple sort in the dimensions. The cardinality of the group in the $i$-th dimension is denoted by $g(v_{1,h_1}|v_{2,h_2}|....|v_{i,h_i})$, where $v_{j,h_j}$ are the corresponding values of the descriptor vectors of the images in dimensions $j = 1, ..., i$ $(1 \leq h_j \leq c'_j)$.

A detailed example is presented in Figure 5, which depicts $N = 44$ images $o_j$, $j = 1, ..., 44$, of $D$ dimensions. For the sake of simplicity, let us examine only the first three dimensions. The first, second and third dimensions have value cardinalities 4 (values $v_{11}, v_{12}, v_{13}, v_{14}$), 3 (values $v_{21}, v_{22}, v_{23}$) and 2 (values $v_{31}, v_{32}$), respectively. Based on the multiple sort algorithm, the images are reorganized by following the hierarchy of equal values, as depicted in Figure 5. Examples of equal value group cardinalities $g$ are: (i) $g(v_{12}) = 11$ (positions from 16 to 26), (ii) $g(v_{14}|v_{21}) = 4$ (positions from 36 to 39), (iii) $g(v_{13}|v_{22}|v_{32}) = 3$ (positions from 31 to 33), (iv) $g(v_{12}|v_{23}|v_{32}) = 2$ (positions 25 and 26).

In the sequel of the analysis, we will focus on the case of the top-1 query $(k = 1)$. However, the analysis can be easily generalized for the case of top-$k$ query $(k > 1)$. Let $o_q$ be the query image, where its descriptor vector has already been indexed into the dataset $\mathbf{S}'$, following the MSIDX insertion algorithm. Let $o_a$ be the top-1 similar image to $o_q$, where, their minimum distance is:

$$d(o_a, o_q) = \min_{o \in \mathbf{S}'} d(o, o_q) \tag{1}$$

Each distance $d$ transposes the above minimization to a system of possible minimizations over the absolute differences in each dimension. For example, if $d$ is calculated by using the $L_2$ distance, then:

$$d(o_a, o_q) = \min_{o \in \mathbf{S}'} d(o, o_q) =$$

$$= \min_{o \in \mathbf{S}'} \sqrt{\sum_{i=1}^{D} (o.x_i - o_q.x_i)^2} \leq \min_{o \in \mathbf{S}'} \sum_{i=1}^{D} |o.x_i - o_q.x_i| \tag{2}$$

To minimize the above sum, the absolute differences $|o.x_i - o_q.x_i|$, $i = 1, ..., D$, have to be minimized, respectively. The best case scenario is when the differences are zero in some dimensions $i$ $(o_a.x_i = o_q.x_i)$, and the remaining differences $|o_a.x_i - o_q.x_i|$ are as minimum as possible. Since in the dimensions, there are relative low value cardinalities (compared to the dataset size, i.e. $c'_i \ll N, \forall i = 1, ..., D$), the probability of having many zero differences is high. This fact constitutes the motivation for performing multiple-sort in the preprocessing algorithm, since the selection of the dimensions priority is based on the descending order of their respective value cardinalities. In particular, the descending order criterion helps in minimizing the cardinalities of the groups $g$, since the hashing of the equality groups $g$ is as maximum as possible and therefore, images with equal values in the first $h$ $(2 \leq h \leq N)$ groups have position-distances as minimum as possible. More specifically, the total number of the groups $g$ that can be defined over the dimensions is:

$$c'_1 + c'_1 \cdot c'_2 + c'_1 \cdot c'_2 \cdot c'_3 + \ldots\ldots = \sum_{i=1}^{D} \prod_{j=1}^{i} c'_j \tag{3}$$

and since it does hold that $c'_1 \geq c'_2 \geq \ldots \geq c'_D$ the above sum is maximized and consequently, the hashing of the groups $g$ is maximum.

Moreover, let $h$ be the dimension $(h \leq D)$ in which the top similar image $o_a$ has the first non-zero difference with the query image $o_q$, while the images $o_a, o_q$ have equal values in the first $h-1$ dimensions and different values in the dimension $h$ according to:

$$\forall i = 1, ..., h-1, \exists j_i \in \{1, ..., c'_i\} : o_a.x_i = o_q.x_i = v_{ij_i}$$

$$\wedge \qquad o_a.x_h \neq o_q.x_h \tag{4}$$

In the rest of the dimensions (next to the $h$-th dimension), the image $o_a$ may have more zero or smaller non-zero differences, but due to the multiple-sort, based on the dimension priority, $o_a$ position-distance from the query image $o_q$ is exclusively dependent on the first $h$ dimensions, according to groups $g$ over the data. More specifically, the position-distance between $o_a$ and $o_q$ (denoted by $d_{pos}$) satisfies the following inequality:

$$d_{pos}(o_a, o_q) \leq g(v_{1,j_1}|v_{2,j_2}| \ldots |v_{h-1,j_{h-1}}) - 1 \tag{5}$$

To ensure that $o_a$ will be retrieved as the top similar image to $o_q$, the value of the parameter $W$ is constrained according to:

$$d_{pos}(o_a, o_q) \leq g(v_{1,j_1}|v_{2,j_2}| \ldots |v_{h-1,j_{h-1}}) - 1 \leq W \tag{6}$$

In the example of Figure 5, the query image $o_q$ and the top similar image $o_a$ have equal values in their first two dimensions, while the values in their third dimension are different $(h = 3)$: $o_a.x_1 = o_q.x_1 = v_{12}$, $o_a.x_2 = o_q.x_2 = v_{23}$ and $o_a.x_3 = v_{32} \neq v_{31} = o_q.x_3$. Therefore, according to (6), the constraints of the $W$ parameter are:

$$d_{pos}(o_a, o_q) \leq g(v_{12}|v_{23}) - 1 \leq W \Leftrightarrow$$

$$\Leftrightarrow W \geq g(v_{12}|v_{23}) - 1 = 5 - 1 = 4 \tag{7}$$

Consequently, the constraint of $W \geq 4$ is necessary, in order to ensure that $o_a$ will be retrieved as the top similar image to $o_q$. For example, in Figure 5, $o_q$ is allocated in position 23 and $o_a$ in position 25 and their position-distance is $d_{pos}(o_a, o_q) = 2 < 4$. Since $o_q$ could be allocated only in positions 22, 23, 24 and $o_a$ only in positions 25, 26 (to comply with the ordering criterion of the multiple sort algorithm), their maximum position-distance cannot exceed $|26 - 22| = 4$.

Since it is infeasible to predefine in which group the top similar image is located, a generalization of (6) follows. More specifically, a general bound of $W$ is calculated by computing the maximum value between the equality group cardinalities $g$ over the dimension $h-1$, according to:

$$W \geq \max_{\substack{1 \leq j_1 \leq c'_1 \\ 1 \leq j_2 \leq c'_2 \\ \cdots\cdots \\ 1 \leq j_{h-1} \leq c'_{h-1}}} \{g(v_{1,j_1}|v_{2,j_2}| \ldots |v_{h-1,j_{h-1}})\} - 1 \tag{8}$$

Therefore, the bound of $W$ is more general and independent from the specific group cardinalities and position allocations

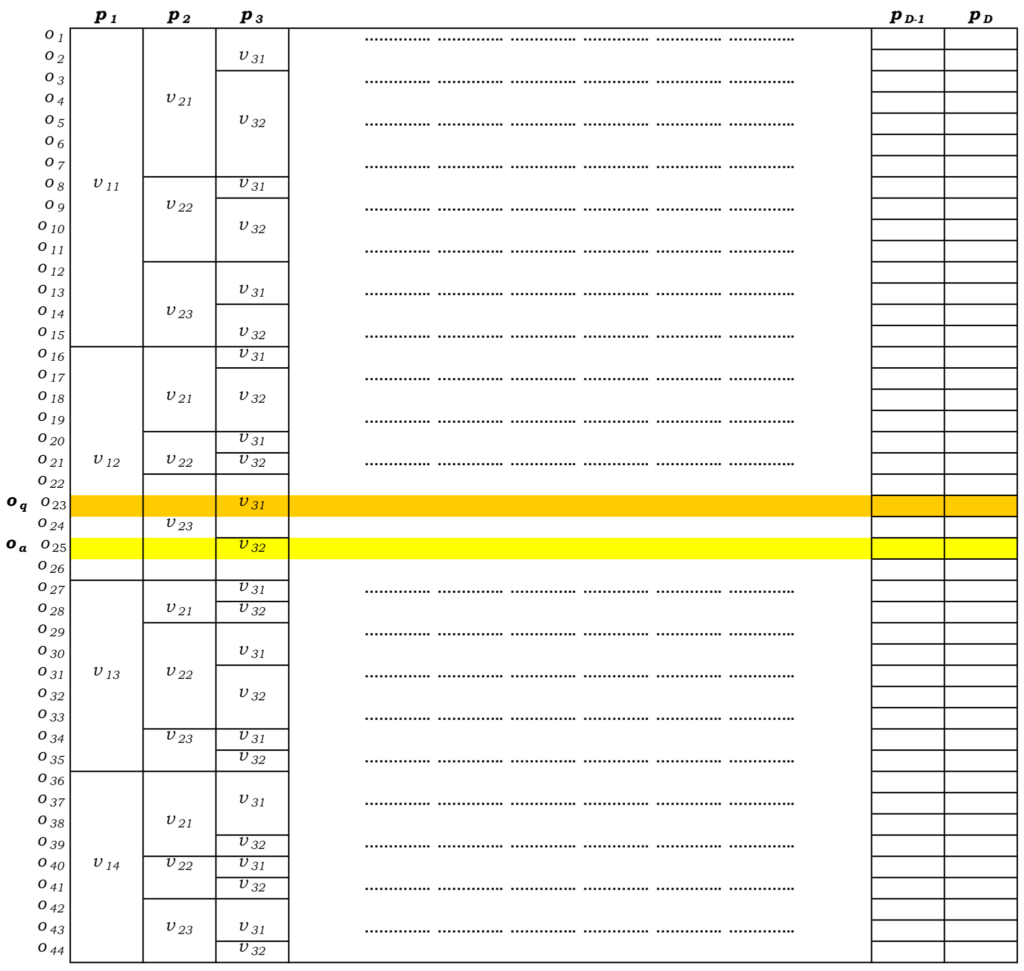


Fig. 5. MSIDX analysis example.

of images $o_a$ and $o_q$. However, the unique constraint is the condition (4), based on which the dimension $h$ has to be predefined.

Let $\overline{g_i}$ be the aforementioned bound of $W$ for each dimension $i = 1, ..., D$. In the example of Figure 5, the general bound of $W$ is calculated according to (8). By considering that $h = 3$ and the equality group cardinalities $g$ of the second dimension are equal to $\{7, 4, 4, 4, 2, 5, 2, 5, 2, 4, 2, 3\}$, then:

$$\overline{g_2} = \max_{\substack{1 \le j_1 \le c'_1 \\ 1 \le j_2 \le c'_2}} \{g(v_{1,j_1} | v_{2,j_2})\} - 1 =$$

$$= \max\{7, 4, 4, 4, 2, 5, 2, 5, 2, 4, 2, 3\} - 1 = 7 - 1 = 6 \quad (9)$$

Therefore, the constraint of $W \geq 6$ must be satisfied in order to ensure that $o_a$ will be retrieved as the top similar image to $o_q$.

Consequently, the $D$ maximum group cardinality values $\overline{g_h}$ over the dimensions $h = 1, ..., D$ are calculated according to (9), constituting the available different lower bounds for the $W$ parameter. Therefore, in the preprocessing algorithm, the calculation of the excepted bounds can be performed for the reordered set **S'** in $O(D \cdot N)$. Additionally, in the insertion algorithm, any new insertion updates instantly the values of $\overline{g_h}$. In the example of Figure 5, the bounds of the parameter $W$ for the first three dimensions according to (9) are: $\overline{g_1} = 14$, $\overline{g_2} = 6$, $\overline{g_3} = 4$, respectively. Thus, all three selections for the $W$ parameter lead the MSIDX query processing to successfully report the top similar image $o_a$.

## B. Probabilistic Analysis

According to the probabilistic analysis that follows, a further generalization of the expected bound for the parameter $W$ is provided, by examining two different cases: (a) the dimensions are statistically independent and the probability distributions of the values in each dimension are predefined and (b) dependencies between the dimensions do exist and the joint probability distribution for all dimensions is predefined. For both cases, the computations of the group cardinalities are not required, since they can be estimated by the probability calculations.

Let $X_i$, $i = 1, ..., D$ be random variables representing the values of the descriptor vectors of the images in each dimension. Following the more general case, in which dependencies between the dimensions do exist, let $f(x_1, x_2, ..., x_D)$ be the *joint distribution function* (joint PDF) of all $X_i$ random variables. Therefore, for a predefined query image $o_q$, the probability of finding a top similar image $o_a$, while complying with the condition (4), is:

$$P(X_1 = v_{1,j_1} \cap X_2 = v_{2,j_2} \cap \ldots \cap X_{h-1} = \\ = v_{h-1,j_{h-1}}) = f(v_{1,j_1}, v_{2,j_2}, \ldots, v_{h-1,j_{h-1}}) \quad (10)$$

where $f(v_{1,j_1}, v_{2,j_2}, \ldots, v_{h-1,j_{h-1}})$ is the corresponding value of the *marginal joint PDF* (note that only $h-1 \leq D$ arguments are considered), which can be derived from the joint PDF. If the variables $X_i$ are continuous, then :

$$f(x_1, x_2, \ldots, x_{h-1}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \\ \cdots \int_{-\infty}^{\infty} f(x_1, x_2, \ldots, x_{h-1}, \ldots)\, dx_h\, dx_{h+1} \ldots dx_D \quad (11)$$

If the variables $X_i$ are discrete, then:

$$f(x_1, x_2, \ldots, x_{h-1}) = \\ = \sum_{x_h} \sum_{x_{h+1}} \cdots \sum_{x_D} f(x_1, x_2, \ldots, x_{h-1}, \ldots) \quad (12)$$

For example, CEDD [9] descriptors contain integers in the range $[0, 7]$, thus the variables $X_i$ are discrete, whereas the GIST [41] descriptors contain real numbers, thus the variables $X_i$ are continuous.

Therefore, the corresponding group cardinality $g$ can be efficiently approximated by the following expected value:

$$g(v_{1,j_1}|v_{2,j_2}|\ldots|v_{h-1,j_{h-1}}) \approx \\ \approx N \cdot f(v_{1,j_1}, v_{2,j_2}, \ldots, v_{h-1,j_{h-1}}) \quad (13)$$

Thus, the expected bound of the parameter $W$ is calculated according to (6), as:

$$W \geq N \cdot f(v_{1,j_1}, v_{2,j_2}, \ldots, v_{h-1,j_{h-1}}) - 1 \quad (14)$$

According to (14), the expected bound of the parameter $W$ is completely independent from the group cardinalities $g$, since it can be calculated immediately from the marginal joint PDF distribution $f$ and the first $h-1$ dimensions of the query image $o_q$. However, in real-world image datasets, despite the fact that the joint PDF distribution is unknown, it can be efficiently approximated following the methodology of [17], but such analysis is out of the scope of this paper.

Moreover, an interesting special case is when the dimensions are statistically independent and the probability distributions of the values in each dimension are predefined. Let $f(x_i)$ be the probability density function of the variable $X_i$ in dimension $i$, for $i = 1, ..., D$. For a predefined query image $o_q$, the probability of finding a top similar image $o_a$, which satisfies the condition of (4), is calculated by the statistical independence between the variables $X_i$ as follows:

$$P(X_1 = v_{1,j_1} \cap X_2 = v_{2,j_2} \cap \ldots \cap X_{h-1} = v_{h-1,j_{h-1}}) = \\ = P(X_1 = v_{1,j_1}) \cdot P(X_2 = v_{2,j_2}) \cdot \ldots \cdot P(X_{h-1} = v_{h-1,j_{h-1}}) = \\ = f(v_{1,j_1}) \cdot f(v_{2,j_2}) \cdot \ldots \cdot f(v_{h-1,j_{h-1}}) = \prod_{i=1}^{h-1} f(v_{i,j_i}) \quad (15)$$

Thus, similar to the previous general case, the expected bound of the parameter $W$ is derived as:

$$W \geq N \cdot \prod_{i=1}^{h-1} f(v_{i,j_i}) - 1 \quad (16)$$

Considering the case that all values are uniformly distributed in each dimension, the expected bound of (16) is:

$$W \geq \frac{N}{c'_1 \cdot c'_2 \cdot \ldots \cdot c'_{h-1}} - 1 \quad (17)$$

This bound is an efficient estimator of the parameter $W$. In the example of Figure 5 ($h = 3$, $c'_1 = 4$, $c'_2 = 3$, $N = 44$), the $W$ bound is calculated according to (17) as :

$$W \geq \frac{44}{4 \cdot 3} - 1 = 2.667 \Leftrightarrow W \geq 3$$

The $W$ parameter verifies in this case as well, that the MSIDX query processing algorithm successfully reports the top similar image.

## V. Experimental Evaluation

In this section, we present experimental results for the proposed MSIDX method, against hashing methods, suitable for approximate similarity search, in terms of search time, retrieval accuracy and preprocessing requirements.

### A. Datasets

The evaluation datasets in our experiments include a wide selection of heterogeneous descriptors extracted from two different image collections, in order to demonstrate the generality of the proposed methodology. All datasets are publicly available and are not subjected to any additional postprocessing, such as quantization or normalization. The first collection is the ImageClef Wikipedia Retrieval 2010 Collection featuring 237,434 (240K) images [61]. Global image descriptors are publicly available for this collection, featuring CIME descriptors [54] (64-dimensions), CEDD descriptors [9] (144-dimensions) and SURF descriptors [3] (5000-dimensions). The second image collection is a combination of the Flickr 1M image collection and the Inria Holidays images [31]. Descriptor datasets are publicly available by the TEXMEX project team (IRISA) [1], [30] featuring two descriptor datasets of 1M images, namely local descriptors SIFT [37] (128-dimensions) and global descriptors GIST [41] (960-dimensions).

CIME [54] (integer value type) is a border/interior classification algorithm which classifies pixels into interior or border and then builds a 64 bins histogram for each pixel type. The feature space is composed of 64 dimensions. The provided descriptor vectors have integer values in the range of [0, 63].

SURF [3] (normalized real value type) is a widely used local descriptor, best known for the combination of its extraction speed and its expressive power. In the dataset of [61], the extracted descriptors from each image are vector-quantized using the bag-of-words methodology [52]. A codebook with a size of 5,000 words is used to map the extracted descriptors to a global histogram. The codebook is created using a random sample of the collection and using K-means clustering to define the words. The descriptor vector is further normalized using the term frequency - inverse document frequency (tf-idf) methodology to enhance the descriptive ability of the descriptor. The provided descriptor vectors have real values due to the normalization but the descriptor dimensions have a finite value cardinality due to the finite value cardinality of the integer values before the normalization. The final value range is [0, 1].

CEDD [9] (integer value type) incorporates color and texture information in a histogram. The image is split into image blocks, which are classified to 6 different texture types. Each texture type is assigned a 24-bin histogram, where the bins are representing a predefined set of colors in the HSV space. Every image block classified to the specific texture type, contributes to its respective histogram. The image block contributes to the respective dominant color bin of the texture type it belongs. Therefore, a 6 x 24 = 144 bin histogram is produced. In order to render the descriptor suitable for large scale image databases and to keep the computational power low, the information is normalized and quantized to limit CEDD's size to 54 bytes per image. Each bin is normalized to the [0, 1] decimal area and quantized to 3bits/bin. The provided descriptor vectors have integer values in the range of [0, 7].

SIFT [37] (integer value type) is among the most popular descriptors. SIFT descriptors were extracted with the Mikolajczyk [39] implementation of Hessian-affine keypoint detector. A histogram of dominant orientations is obtained by considering pixels around a radius of each keypoint. SIFT descriptors are robust to local affine distortion. The extracted descriptors have a dimension of 128 bins. The provided descriptor vectors have integer values in the range of [0, 255].

GIST [41] (real value type) is a descriptor that tries to model the structure of the scene using a set of perceptual dimensions (naturalness, openness, roughness, expansion, ruggedness) that represent the dominant spatial structure of a scene. These dimensions may be reliably estimated using spectral and coarsely localized information. The descriptor vectors have real values, that are further normalized, divided by the local luminance variance. The provided descriptor vectors have 960 dimensions of real values in the range of [0, 1.0929].

We denote the datasets from the ImageClef Wikipedia Retrieval 2010 Collection as: CIME-64d, CEDD-144d, SURF-5000d, and the datasets from the TEXMEX project as: SIFT-128d, and GIST-960d. For each dimension of the datasets, the observed value cardinalities are depicted in Figure 6. The observed value cardinalities in the datasets with descriptor vectors of integer values (CIME-64d, CEDD-144d, SIFT-128d) are extremely low in comparison with the dataset size (240K for CIME and CEDD, 1M for SIFT). For CEDD-144d and CIME-64d the cardinality is bound to 8 and 64 respectively due to the extraction process of the descriptor. For CIME-64d the maximum observed value cardinality is 57 and the mean is 46. For CEDD-144d the maximum observed value cardinality is 8 and the mean is 7.2. For SIFT-128d the maximum observed value cardinality is 210 and the mean is 168.5. Moreover, the observed value cardinalities in the datasets with descriptor vectors of real values (SURF-5000d, GIST-960d) are also relative low in comparison with the corresponding dataset size (240K for SURF and 1M for GIST). More specifically, in SURF-5000d the maximum observed value cardinality is 20,958 (since the second max value is 4,088, the respective figure is bounded to 5,000 for illustration purposes) and the mean is 896.5. Finally, in GIST-960d the maximum observed value cardinality is 5,664 and the mean is 3,599.1. Therefore, in all five evaluation datasets we observe that there are few dimensions with high discriminative power (i.e. those that have high value cardinalities), while the rest of dimensions have low value cardinalities, compared to the dataset size.

## B. Experiments Organization and Protocols

Following the evaluation protocol of [25] and [33], in all our experiments we performed 1,000 test queries, which were randomly chosen and did not participate into the training/preprocessing phase. For each query, the retrieval accuracy is measured according to the following ratio:

$$\text{Accuracy } = \frac{|\mathbf{R}_{ex} \cap \mathbf{R}|}{k} \cdot 100\% \tag{18}$$

where, $\mathbf{R}_{ex}$ is the set of the top-$k$ results (nearest neighbor items) retrieved by the exhaustive search (ground truth), based on the Euclidean distance ($L_2$), and $\mathbf{R}$ is the set of the top-$k$ results retrieved by the tested method. The final performance of each method is measured by the mean Average Precision (mAP), which is defined as the average retrieval accuracy of the 1,000 performed queries.

For the data-dependent hashing methods, we used as a training set all images from the original dataset, except those included in the test queries (following the methodology of [21]). Moreover, for the data-dependent hashing methods that support multiple hash tables, we also averaged over five random training and test partitions.

The proposed method was implemented in C++ and it is available at [62]. All experiments were conducted in a machine of 3GHz CPU with 8GB main memory, running Windows 7 64-bit.

Since the proposed method significantly differs from all hashing methods, an additional rule is used in the evaluation protocol, that deals with the query response time. The additional rule is that: "if a parameter setup in a tested method leads to query response time greater than the search time of exhaustive search, then the parameter setup is considered completely inefficient and thus, further tuning is avoided". The performance of each tested method is measured by the *search time*, which is defined as the average query response time of the 1,000 performed queries. In the case of the hashing methods that support multiple hash tables, we varied the number of hash tables, to achieve the maximum retrieval accuracy, keeping the search time lower than the respective search time of exhaustive search. For all hashing methods we also varied the number of bits in the binary encodings, preserving compliance with the aforementioned rule.

## C. State-of-the-art Hashing Methods

We compare our proposed MSIDX method against the following hashing methods:

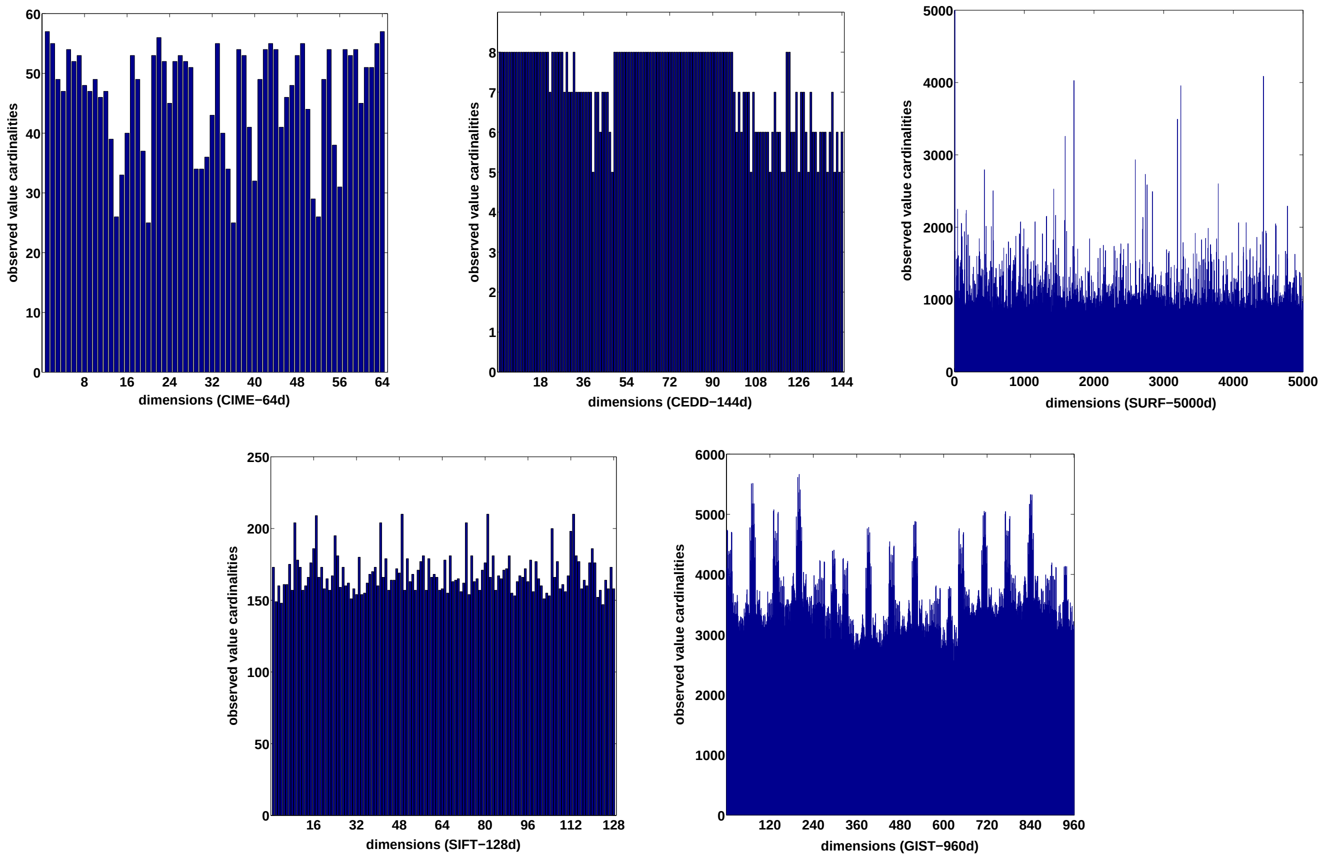


Fig. 6. Observed value cardinalities of the dimensions in the evaluation datasets.

- **LSH:** Locality Sensitive Hashing [19], which projects items through a gaussian random distribution. We used the same implementation of LSH and parameters setup that were used in [25]. The source code is publicly available at [26].
- **SPH**: Spherical Hashing [25], which is a recent hashing method, and it is based on a hypersphere binary embedding technique. Between the two proposed variants, Hamming Distance (HD) based and Spherical Hamming Distance (SHD) based, we used the SHD in SPH, because based on the experimental evaluation of [25], SHD achieves high retrieval accuracy in terms of mAP. We used the same implementation and parameters setup that were used in [25]. The source code is provided by the authors of [25] and is publicly available at [26].

We also compare the proposed MSIDX method to several state-of-the-art hashing methods for the dataset GIST-960d, using the same parameter setups as presented in the experimental evaluation of [25]:

- **LSH-ZC**: A variant of Locality Sensitive Hashing with Zero Centered data points [15].
- **PCA-ITQ**: Iterative Quantization [20].
- **GSPICA-RBF**: Generalized Similarity Preserving Independent Component Analysis with the RBF kernel [24].
- **RMMH**: Random Maximum Margin Hashing [33].
- **LSBC**: Locality Sensitive Binary Codes [45].
- **SpecH**: Spectral Hashing [58].

## *D. Impact of the $W$ Parameter*

In MSIDX, the $W$ parameter defines a search radius from the query image over the indexed images. Therefore, the search window has a maximum width of $2W$. Moreover, if the $W$ parameter is close to $0.50$ ($50\%$ of the dataset size $N$), then MSIDX performs similar to the exhaustive search algorithm, achieving accuracy equal to $100\%$. Additionally, several experiments were conducted to examine possible aggregation measures and priority indices, corresponding to the optional value $ex$ in the extra dimension. Therefore, we concluded to the norm square of the image based on the following formula: $ex = ||o_j||^2 = \sum_{i=1}^{D} x_{ij}^2$, in order to achieve an additional $4\%$ mAP increase in all datasets.

In Figure 7, we present the experimental results for evaluating the performance of MSIDX by varying the $W$ parameter. More specifically, we performed top-$k$ nearest neighbor search in the 5 different datasets for $k = 100$, and we observed that mAP increases significantly with respect to $W$, almost following a logarithmic aggression. Note that the $x$-axis of the graphs in Figure 7 is in logarithmic scale, and the observed points (after the second value of $W$) are almost collinear. The required exhaustive search time is depicted as a perpendicular line for each dataset in the Figures. For $W$ close to $0.05$ (only $5\%$ of $N$), a mAP of over $30\%$ is achieved in all datasets.

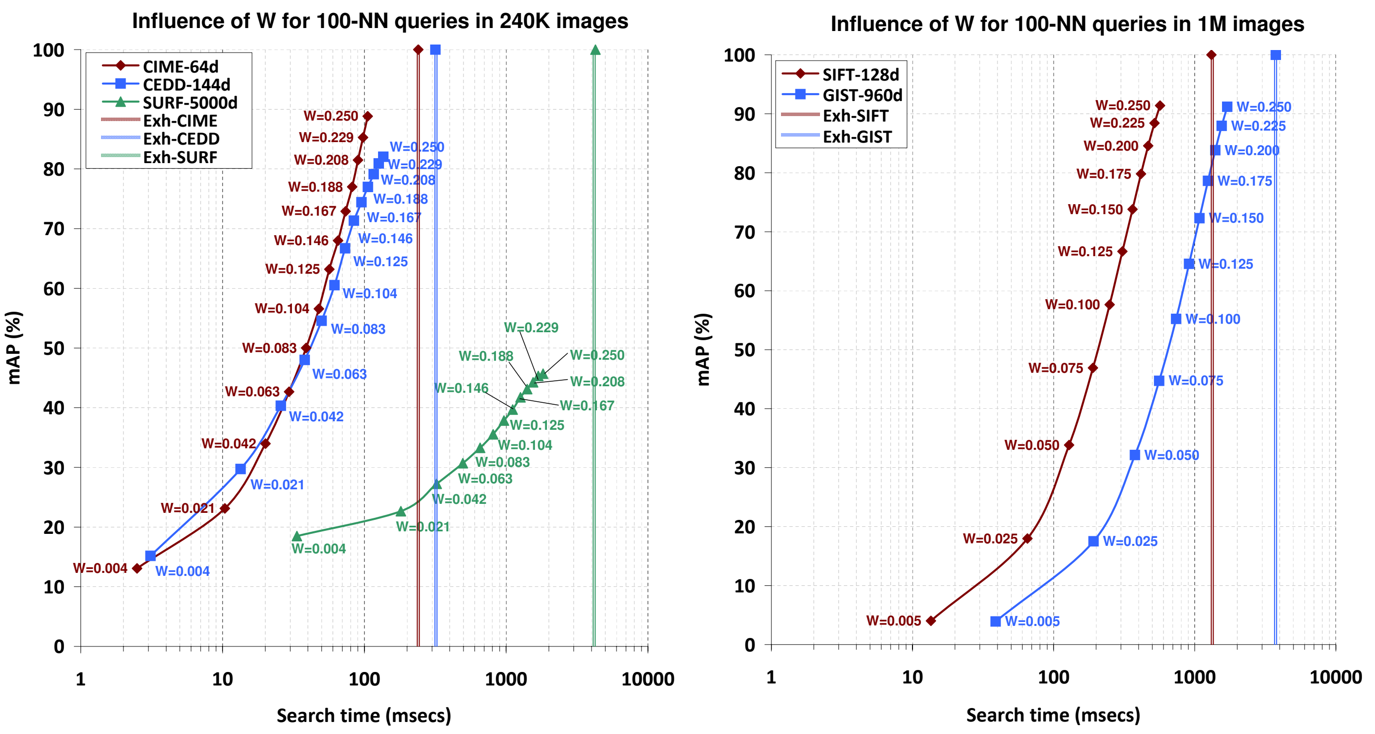


Fig. 7. Influence of the $W$ parameter for 100-NN queries.

This is an important result because with a significantly small search time a relative high mAP is achieved. As we present later, none of the state-of-the-art hashing methods have similar performance. If $W$ is located between $0.15$ and $0.20$, then a high mAP (over $70\%$) is achieved in all datasets, except SURF-5000d which returns mAP between $40\%$ and $50\%$. Finally, for all datasets (except SURF-5000d), if $W$ equals $0.25$, then the achieved mAP is close to $90\%$, and thus a further increase of $W$ over $0.25$ is not required.

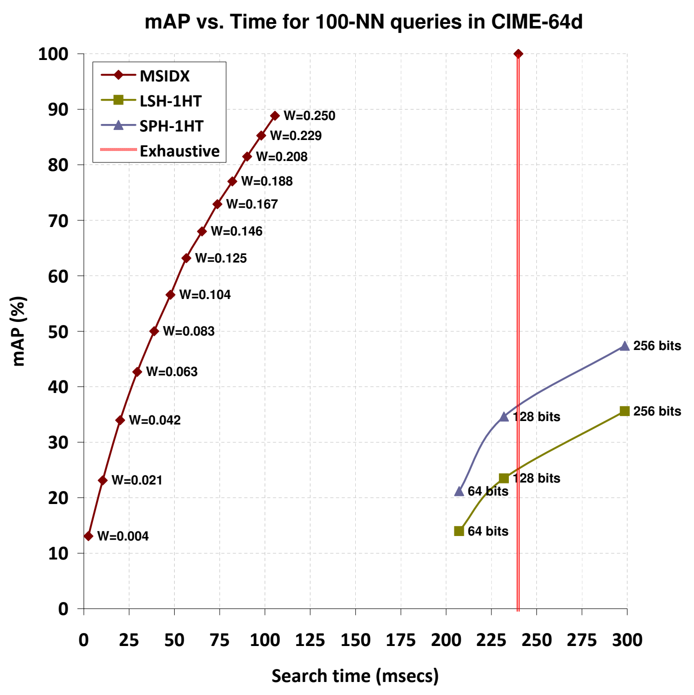
## *E. Comparison against state-of-the-art Hashing Methods*

In the first set of experiments we evaluate MSIDX against (a) LSH (as a baseline method) and (b) SPH, which outperforms state-of-the-art hashing methods, as already described in Section II-C, and shown in [25]. In Figure 8, we present the experimental results by performing 100-NN queries for CIME-64d, CEDD-144d, SIFT-128d, SURF-5000d. Since GIST-960d is also used in the experimental evaluation of SPH, we performed 1000-NN queries, following the parameter setup of [25]. In LSH and SPH we vary the number of bits in the binary codes, and the number of the hash tables to achieve the maximum retrieval accuracy, while always preserving the search time less than the respective search time of exhaustive search.

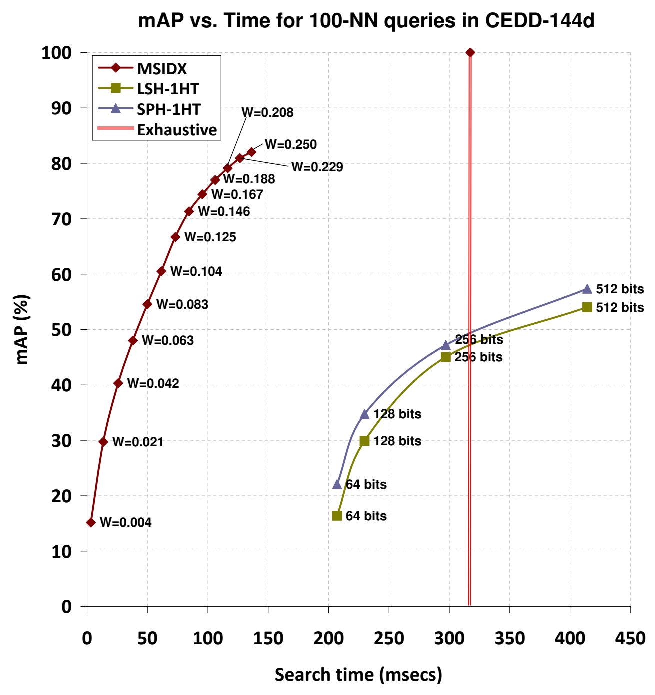
In CIME-64d, CEDD-144d, and SIFT-128d, the maximum allowed number of hash tables for LSH and SPH is one (denoted as 1HT), because a second hash table leads to a search time larger than that of the exhaustive search. Note that the required exhaustive search time is depicted as a perpendicular red line for each dataset in the Figures. Moreover, the maximum allowed number of bits in the binary codes of LSH and SPH is 128 and 256, because a further increase leads to search time larger than that of the exhaustive search. As expected, SPH outperforms LSH in all cases, except for the case of SIFT-128d with 64 and 128 bits code length, where SPH appears to have similar performance with LSH. However, in all cases MSIDX clearly outperforms both LSH and SPH in terms of mAP and search time. More specifically, MSIDX achieves a mAP close to $90\%$ with significantly less search time than that of LSH and SPH, even if 64 bits are used in the binary encodings. Moreover, LSH and SPH achieve a mAP less than $40\%$, $50\%$, $20\%$ in CIME-64d, CEDD-144d, and SIFT-128d respectively, for the maximum allowed number of bits. Finally, an important result is that the limited retrieval accuracy of LSH and SPH is overcomed by MSIDX, even for low values of $W$.

In SURF-5000d the required search time for exhaustive search is high due to the existence of high dimensionality. Thus, the maximum allowed number of hash tables for LSH and SPH are set to 14. In all cases, MSIDX outperforms SPH and LSH. An interesting observation is that SPH achieves almost the same maximum mAP as MSIDX (for 256 bits code length and 11 hash tables or for 128 bits code length and 14 hash tables). However, SPH requires a search time over 3,900 msec (close to that of the exhaustive search), whereas MSIDX for the same mAP requires only 1,800 msec. The maximum mAP of LSH remains close to $30\%$ (for 512 bits code length and 8 hash tables). Finally, an important result is that in all cases the retrieval accuracy of SPH is exceeded by MSIDX, for significantly lower search times.

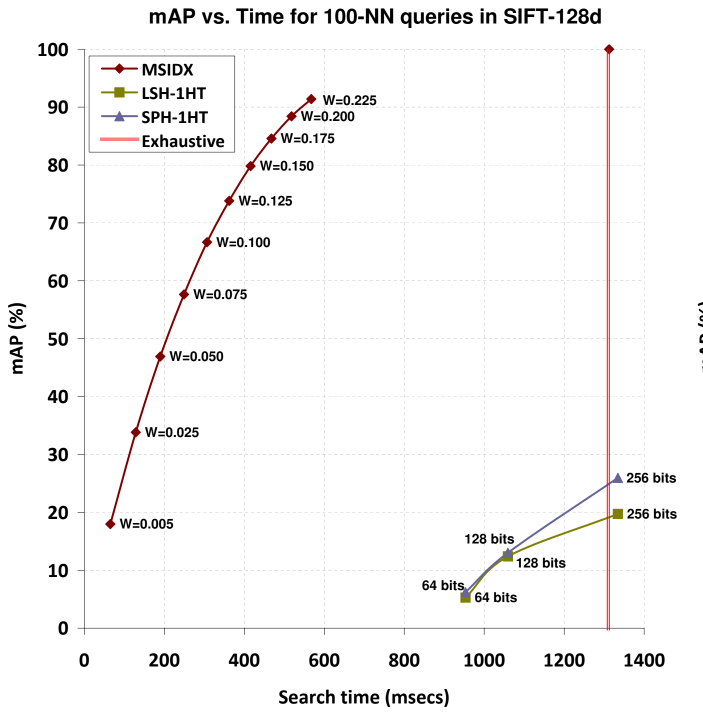
In GIST-960d the required search time for exhaustive search is also high, due to the existence of high dimensionality. Thus,



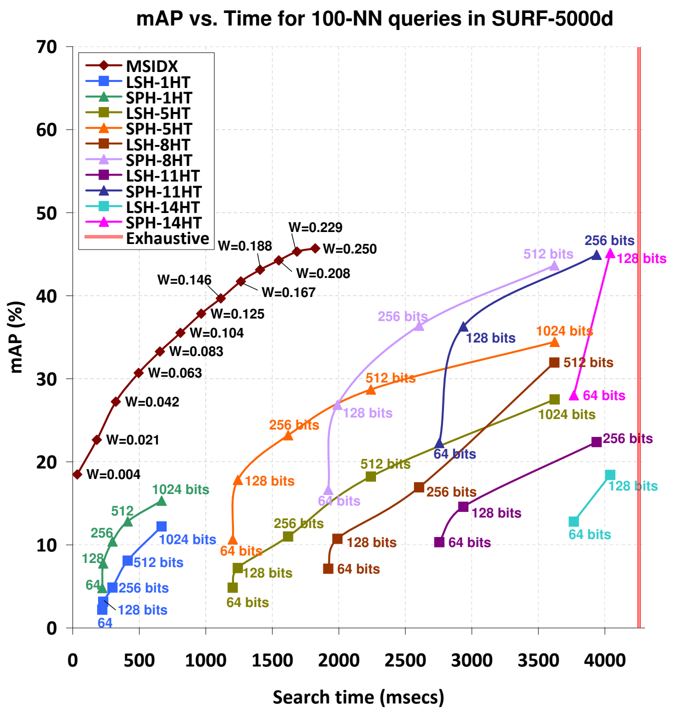
(a)



(b)



(c)

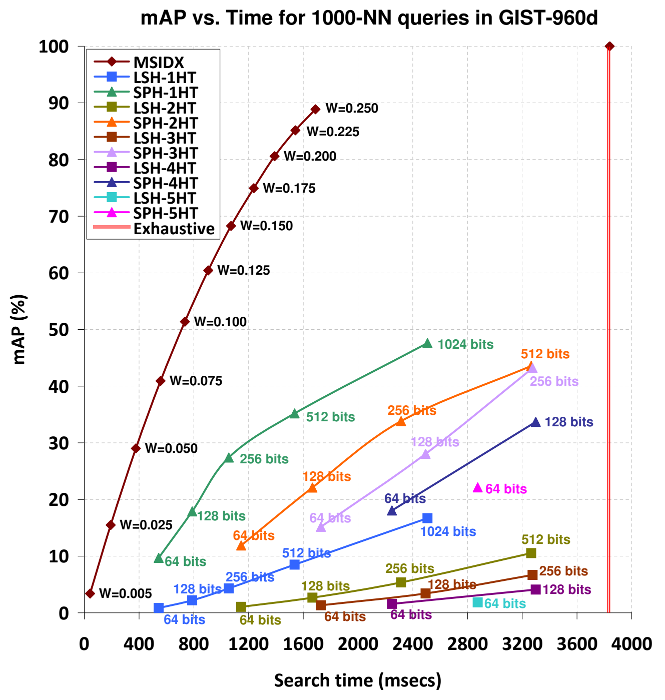


(d)

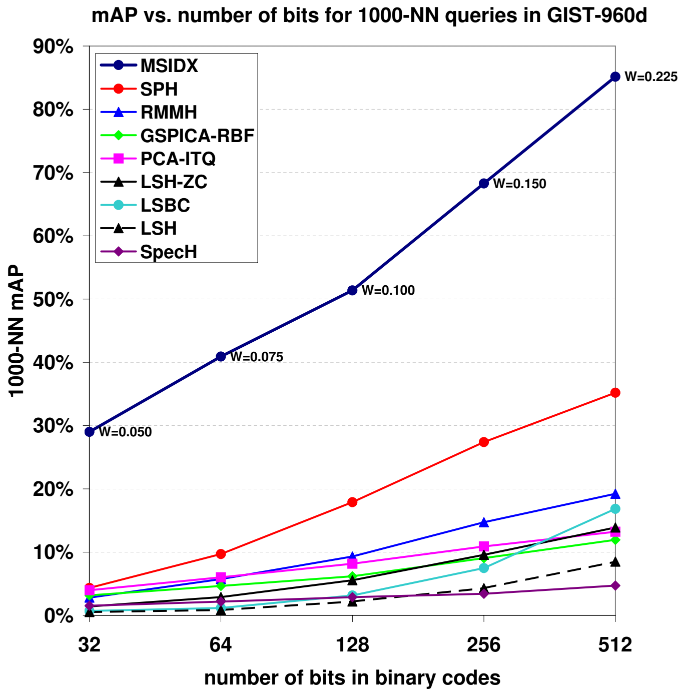
Fig. 8. Retrieval accuracy versus search time for 100-NN queries in : (a) CIME-64d, (b) CEDD-144d, (c) SIFT-128d, and (d) SURF-5000d.

the maximum allowed number of hash tables for LSH and SPH is 5. In all cases, MSIDX significantly outperforms SPH and LSH. SPH achieves a maximum mAP close to 50% (for 1,024 bits code length and 1 hash table), while LSH achieves a maximum mAP close to $20\%$ for the same settings. Clearly, the retrieval accuracy of LSH and SPH is overcomed by MSIDX, for low values of $W$ and lower search time.

In the next set of experiments we further compare MSIDX against state-of-the-art hashing methods in the GIST-960d dataset, following the presentation scheme of [25]. In Figure 9b we show the experimental results of 1,000-NN queries in the same bit range for the binary encodings that are also used in [25]. In this experiment, the number of bits varies (and not the search time), thus for MSIDX we use the corresponding $W$



(a)



(b)

Fig. 9. Retrieval accuracy for 1,000-NN queries in GIST-960d versus: (a) search time, and (b) number of bits as presented in [25].

values that produce similar search times with SPH in the selected bit range. More specifically, for 32 bits code length SPH requires 382.14 msec, whereas MSIDX requires 376.69 msec for $W = 0.050$. For 64, 128, 256, 512 bits code length SPH requires 543.31, 788.64, 1055.92, 1536.43 msec, respectively, whereas for $W = 0.050, 0.075, 0.100, 0.150, 0.225$, MSIDX requires 556.95, 734.81, 1072.14, 1542.37 msec, respectively. We observe that MSIDX significantly outperforms the hashing methods in all cases, by returning for the same search time, a minimum mAP close to $30\%$ and a maximum mAP over $80\%$.

### *F. Preprocessing Requirements*

In the last set of experiments we report the preprocessing time requirements for MSIDX, SPH and LSH. Note that in the requirements we do not include any I/O cost for loading or managing the images in the disk level, considering only the required CPU time for preprocessing, after loading all images in main memory. More specifically:

- For MSIDX we recorded the CPU time that is required to perform the preprocessing algorithm, described in Section III-A.
- For SPH we recorded the CPU time that is required for learning the spherical hashing plus the CPU time for computing the binary codes.
- For LSH we recorded the CPU time that is required for computing the binary codes.

In Table I the respective results are depicted. We must note that these results are in case of one hash table. In case that more hash tables are used, each table requires a similar preprocessing cost. We can observe that (a) SPH requires significantly higher preprocessing time than LSH, because the former has the additional time/cost for learning the spherical hashing, (b) MSIDX requires significantly lower preprocessing time than SPH and LSH in all cases, by performing the preprocessing algorithm, described in Section III-A and thus avoiding to compute the binary codes, as it happens in the case of the hashing methods.

## VI. Conclusion

MSIDX is a promising indexing scheme, which analyzes the image content according to the value cardinalities that appear on the dimensions of the respective descriptor vectors. The proposed scheme supports the desired functionalities of modern applications, since it is capable of performing accurate content-based retrieval in low search time and handles the dynamic operations of insertions and deletions in real-time. Through extensive experimental evaluation of MSIDX in five different collections of image descriptor vectors, we showed the superiority of the proposed indexing scheme against other state-of-the-art hashing methods, also suitable for approximate similarity searh. In particular, we illustrated how MSIDX preserves low search time, by also shattering the glass ceiling of the hashing method's limited accuracy. The proposed method is affected by the $W$ parameter, since a trade-off between accuracy and search time does exist, where the increase of $W$ results in "paying" more search time in order to achieve higher accuracy. Since the $W$ parameter is crucial for the performance of MSIDX, a detailed deterministic and probabilistic analysis was provided to set the accepted bounds of $W$. Furthermore, based on the experimental evaluation, we verified that high

| | CIME-64d | CEDD-144d | SIFT-128d | GIST-960d | SURF-5000d |
|---|---|---|---|---|---|
| MSIDX | 3.900 | 4.212 | 20.280 | 24.679 | 29.934 |
| LSH (64 bits) | 6.146 | 12.595 | 47.142 | 656.025 | 884.374 |
| LSH (128 bits) | 12.495 | 25.117 | 94.824 | 1256.359 | 1932.208 |
| LSH (256 bits) | 25.109 | 50.331 | 189.993 | 2803.398 | 4244.286 |
| LSH (512 bits) | 50.108 | 100.626 | 380.698 | 6824.068 | 11494.833 |
| LSH (1024 bits) | 100.817 | 201.461 | 765.005 | 17672.179 | 27554.423 |
| SPH (64 bits) | 75.499 | 104.712 | 114.371 | 1149.372 | 3263.955 |
| SPH (128 bits) | 164.129 | 257.684 | 243.735 | 2350.713 | 7041.005 |
| SPH (256 bits) | 371.336 | 534.758 | 510.681 | 4669.377 | 13759.609 |
| SPH (512 bits) | 837.145 | 1168.819 | 1103.283 | 9405.824 | 28129.575 |
| SPH (1024 bits) | 2020.495 | 2638.818 | 2402.607 | 19186.035 | 58079.639 |

TABLE I
PREPROCESSING TIME REQUIREMENTS (IN SEC).

accuracy is achieved, even for small values of $W$ and a further increase of $W$ is not required.

## ACKNOWLEDGEMENT

We would like to express our gratitude to J. P. Heo, Y. Lee, J. He, S. Chang, and S. Yoon for providing us the source code of Spherical Hashing.

This work was supported by the EC FP7 funded project CUBRIK, ICT- 287704 (www.cubrikproject.eu).

## REFERENCES

[1] L. Amsaleg, H. J'egou, "TEXMEX: Datasets for approximate nearest neighbor search", http://corpus-texmex.irisa.fr/, 2010.

[2] M. Balasubramanian, E.L. Schwartz, J.B. Tenenbaum, V. de Silva, and J.C. Langford, "The ISOMAP algorithm and topological stability", *Science*, vol. 295, no. 5552, pp.7, 2002.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features", *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346-359, 2008.

[4] M. Belkin, and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation", *Neural Comput.*, vol. 15, no. 6, pp. 1373-1396, 2003.

[5] J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, 1975.

[6] J. L. Bentley, "Multidimensional binary search trees in database applications", *IEEE Trans.on Software Engineering*, vol. 5, no. 4, pp. 333-340, 1979.

[7] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful?", in *Proc. of ICDT*, pp. 217-235, 1999.

[8] C. Bohm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases.", *ACM Computing Survey*, vol. 33, no. 3, pp. 322-373, 2001.

[9] S. A. Chatzichristos and Y. S. Boutalis, "CEDD: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval", in *ICVS, volume 5008 of Lecture Notes in Computer Science, Springer*, pp. 312-322, 2008.

[10] E. Chavez, J. Marroquin, and G. Navarro, "Overcoming the curse of dimensionality", in *Proc. of European Workshop on Content-Based Multimedia Indexing (CBMI)*, pp. 57-64, 1999.

[11] E. Chavez, G. Navarro, R. Baeza-Yates and J. L. Marroquin, "Searching in metric spaces", *ACM Computing Surveys*, vol. 33, no. 3, pp. 273-321, 2001.

[12] T. Chiueh, "Content-based image indexing", in *Proc. of Int. Conf. on Very Large Databases (VLDB)*, pp. 582-593, 1994.

[13] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting", in *Proc. of British Machine Vision Conference*, 2008.

[14] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces", in *Proc. of Int. Conf. on Very Large Databases (VLDB)*, pp. 426-435, 1997.

[15] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions", in *Proc. of Symposium on Computational Geometry*, pp. 253-262, 2004.

[16] F. Dehne and H. Nolteimer, "Voronoi trees and clustering problems", *Information Systems*, vol. 12, no. 2, pp. 171-175, 1987.

[17] A. Dvoretzky, J. Kiefer, J. Wolfowitz, "Asymptotic Minimax Character of the Sample Distribution Function and of the Classical Multinomial Estimator", *Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 642-669, 1956.

[18] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time", *TOMS*, vol. 3, no. 3, pp. 209-226, 1977.

[19] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing", in *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pp. 518-529, 1999.

[20] Y. Gong, and S. Lazebnik, "Iterative quantization: a procrustean approach to learning binary codes", in *Proc. of CVPR*, pp. 817-824, 2011.

[21] Y. Gong, S. Lazebnik, A.Gordo, and F. Ferronnin"Iterative quantization: a procrustean approach to learning binary codes for large-image retrieval", *IEEE Trans. on PAMI*, to appear.

[22] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching", in *Proc. of ACM SIGMOD*, pp. 47-57, 1984.

[23] X. He, and P. Niyogi, "Locality preserving projections", in *Proc. of NIPS"*, 2003.

[24] J. He, R. Radhakrishnan, S.-F. Chang, and C. Bauer, "Compact hashing with joint optimization of search accuracy and time", in *Proc. of CVPR*, pp. 753-760, 2011.

[25] J. P. Heo, Y. Lee, J. He, S. Chang, and S. Yoon, "Spherical Hashing", in *Proc. of CVPR*, pp. 2957-2964, 2012.

[26] J. P. Heo, Y. Lee, J. He, S. Chang, and S. Yoon, "Spherical Hashing Source Codes", sglab.kaist.ac.kr/Spherical_Hashing/, 2012.

[27] Z. Huang, H. T. Shen, J. Shao, S. M. Ruger, and X. Zhou, "Locality condensation: a new dimensionality reduction method for image retrieval", in *Proc. of ACM Multimedia*, pp. 219-228, 2008.

[28] Z. Huang, H. T. Shen, J. Liu, and X. Zhou, "Effective data co-reduction for multimedia similarity search", in *Proc. of ACM SIGMOD*, pp. 1021-1032, 2011.

[29] P. Indyk, and R. Motwani,"Approximate nearest neighbors: toward removing the curse of dimensionality", in *Proc. of STOC*, 1998.

[30] H. Jegou, M. Douze and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search", in *Proc. European Conference on Computer Vision, volume I of LNCS, Springer*, pp. 304-317, 2008.

[31] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search", *IEEE Trans. on PAMI*, vol. 33, no. 1, pp. 117-128, 2011.

[32] A. Joly, and O. Buisson, "A posteriori multi-probe locality sensitive hashing", in *Proc. of ACM Multimedia*, pp. 209-218, 2008.

[33] A. Joly, and O. Buisson, "Random Maximum Margin Hashing", in *Proc. of CVPR*, pp. 873-880, 2011.

[34] A. Joly, C. Frelicot, and O. Buisson, "Feature statistical retrieval applied to content-based copy identification", in *Proc. of Int. Conf. on Image Processing*, pp. 681-684, 2004.

[35] W. Johnson, and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space", *Contemporary mathematics*, vol. 26, pp. 189206, 1984.

[36] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search", in *Proc. of IEEE Inter. Conf. on Computer Vision (ICCV)*, pp. 2130-2137, 2009.

[37] D. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, vol. 60, pp. 91-110, 2004.

[38] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search", in *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pp. 950-961, 2007.

[39] K. Mikolajczyk and C. Schmid, "Scale and Affine invariant interest point detectors", *Int. Journal in Computer Vision*, vol. 60, no. 1, pp. 63-86, 2004.

[40] G. Navarro, "Searching in metric spaces by spatial approximation", *The VLDB Journal*, vol. 11, no. 1, pp. 28-46, 2002.

[41] A. Oliva and A. Torralba, "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope". *Int. J. Comput. Vision*, vol 42, no. 3, pp. 145-175, 2001.

[42] L. Pauleve, H. Jegou, and L. Amsaleg, "Locality sensitive hashing: A comparison of hash function types and querying mechanisms". *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1348-1358, 2010.

[43] R. Panigrahy, "Entropy based nearest neighbor search in high dimensions", in *Proc of SODA*, pp. 1186-1195, 2006.

[44] S. Poullot, O. Buisson, and M. Crucianu, "Z-grid-based probabilistic retrieval for scaling up content-based copy detection", in *Proc. of ACM Int. Conf. on Image and Video Retrieval (CIVR)*, pp. 348-355, 2007.

[45] M. Raginsky, and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels", in *Proc. of NIPS*, pp. 1509-1517, 2009.

[46] S. T. Roweis, and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding", *Science*, vol. 290, pp. 2323-2326, 2000.

[47] H. Samet, "The quadtree and related hierarchical data structures", *ACM Computing Surveys*, vol. 16, no. 2, pp. 187-260, 1984.

[48] H. Samet, "Foundations of Multidimensional and Metric Data Structures", Morgan Kaufmann, 2006.

[49] K.E.A. Van De Sande, T. Gevers, and C.G.M. Snoek, "Evaluating Color Descriptors for Object and Scene Recognition", *IEEE Trans. on PAMI*, vol.32, no.9, pp.1582-1596, 2010.

[50] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest Neighbor Methods in Learning and Vision: Theory and Practice*, MIT Press, 2006.

[51] H. T. Shen, X. Zhou, and A. Zhou, "An adaptive and dynamic dimensionality reduction method for high-dimensional indexing", *The VLDB Journal*, vol. 16, no. 2, pp. 219-234, 2007.

[52] J. Sivic, A. Zisserman, "Video Google: Efficient Visual Search of Videos", Toward Category-Level Object Recognition, Lecture Notes in Computer Science Series, Springer, pp. 127-144, 2006.

[53] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval", *in Proc. of ACM Multimedia*, pp. 423-432, 2011.

[54] R.O. Stehling, M. A. Nascimento, and A. X. Falcao, "A compact and efficient image retrieval approach based on border/interior pixel classification", in *Proc. of CIKM*, 2002.

[55] E. Vidal, "An algorithm for finding nearest neighbors in (approximately) constant average time.", *Pattern Recognition Letters*, vol. 4, pp. 145-157, 1986.

[56] J. Wang, S. Kumar, and S.-F. Chang, "Semisupervised hashing for scalable image retrieval", in *Proc. of CVPR*, pp. 3424-3431, 2010.

[57] R. Weber, H. J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces", in *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pp. 194-205, 1998.

[58] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing", in *Proc. of NIPS*, pp. 1753-1760, 2008.

[59] M. Wichterich, I. Assent, P. Kranen, and T. Seidl, "Efficient EMD-based similarity search in multimedia databases via flexible dimensionality reduction", in *Proc. of ACM SIGMOD*, pp. 199-212, 2008.

[60] J. Yang , Y. G. Jiang , A. G. Hauptmann and C. W. Ngo "Evaluating bag-of-visual-words representations in scene classification", in *Proc. of ACM MIR*, pp. 197-206, 2007.

[61] "The ImageCLEF 2010 Wikipedia image collection", http://www.imageclef.org/wikidata, 2010.

[62] http://vcl.iti.gr/msidx/, 2012.

**Eleftherios Tiakas** was born in Thessaloniki, Greece, in 1972. He received the Diploma degree in Mathematics (1994), the Diploma degree in Computer Science Informatics (2006), the M.Sc. degree in Information Systems (2006), and the Ph.D. degree in Computer Science (2011) from the Departments of Mathematics and Informatics of Aristotle University of Thessaloniki, Greece. His research interests include: Spatial & Spatio-Temporal Databases, Multimedia Databases, Data Mining, Information Retrieval, Algorithm Design and Analysis, Graph Theory, Discrete Mathematics, Bio-Informatics, Computational Geometry. Currently, he is a Post-doc Researcher at the Department of Informatics of Aristotle University of Thessaloniki.

**Dimitrios Rafailidis** was born in Larisa, Greece, in 1982. He received the Diploma in Informatics from the Computer Science Department, the M.Sc degree in Information Systems and the Ph.D. degree in Music Information Retrieval from the Aristotle University of Thessaloniki, Greece in 2005, 2007 and 2011, respectively. His main research interests include data mining, information retrieval and recommender systems. Currently, he is a Postdoctoral Research Fellow at Information Technologies Institute of the Center for Research and Technology Hellas (CERTH).

**Anastasios Dimou** is an Electrical and Computer Engineer (diploma 04, Aristotle University of Thessaloniki, Greece; PDEng in Information and Communication Technology, 07 TU Eindhoven, the Netherlands). Since February 2008, he has been a Research Assistant at the Information Technologie Institute / Centre for Research and Technology, Hellas (ITI/CERTH). His research interests focus on multimedia signal processing and analysis, content-based information retrieval, video coding, smart surveillance systems and biomedical applications.

**Petros Daras** (M'07) was born in Athens, Greece, in 1974. He received the Diploma degree in electrical and computer engineering, the M.Sc. degree in medical informatics, and the Ph.D. degree in electrical and computer engineering, all from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1999, 2002, and 2005, respectively. He is a Researcher Grade C, at the Information Technologies Institute of the Centre for Research and Technology Hellas (CERTH). His main research interests include search, retrieval and recognition of 3-D objects, 3-D object processing, medical informatics applications, medical image processing, 3-D object watermarking, and bioinformatics. He serves as a reviewer/evaluator of European projects. Dr. Daras is a Chair of the IEEE MMTC Interest Group (IG) of Image, Video and Mesh Coding and a key member of the IEEE MMTC 3DRPC IG.