Global Scientific JOURNALS

GSJ: Volume 7, Issue 4, April 2019, Online: ISSN 2320-9186
www.globalscientificjournal.com

PERFORMANCE EVALUATION OF GSPEBH BASED ON MEAN AVERAGE PRECISION

Boukari Souley, Abubakar Usman Othman, Abdulsalam Ya'u Gital,Iliya Musa Adamu

Abubakar Tafawa Balewa University, Bauchi, Nigeria
Othman Email: othman80s@yahoo.com

Abstract

The rapid growth in the volume of data has resulted to what is known as big data. This data are indexed for efficient retrieval in the cloud environment. Recently, attention of researchers is drawn to designing efficient indexing techniques searching schemes in the field of data mining, machine learning, pattern recognition, and object matching. ANN methods is needed for similarity search to reduce the computational complexity and memory requirements as against the straight forward techniques that uses exhaustive comparison. Hashing based indexing algorithms are gaining much attention due to their memory efficiency and search accuracy. However, the use of selective projections generation has shown to be inefficient due to the long code length. For an efficient indexing algorithm, we propose a Geo-similarity preserving embedding-based hashing algorithm for improving the search accuracy and memory cost. Extensive experiment is carried out GIST 1M dataset and evaluate the performance of our method and compared our result based on the mean average precision. The results shows that our method outperform-state-of-the-art techniques.

Keywords

Approximate nearest neighbours, indexing, similarity preserving, search accuracy, memory cost.

1 Introduction

With the advent of internet, couple with the availability of hand held devices such as smart phones [1] has contributed to the large volume of data or big data, considering their broad range of resources [2] in the recent years. Millions to billions to trillions of information have been uploaded to more than 300 million World Wide Web accessible websites [3]. This information are continuous in circulation through websites such as Flickr, Twitter, You Tube and yahoo etc. For instance, millions of messages are exchange daily through yahoo, thousands of photos are shared through flickr, millions of tweets on a daily basis and uploading of millions of videos through You Tube. The existing technological infrastructure do not meet the ever heterogeneous and complex growth of data generated and stored in the database. Information extraction is necessary to structuralize the raw data so that they can

be easily digested by analysis algorithms [4]. Retrieving these information from the database becomes a challenging task. For these, efficient indexing techniques are required for efficient storage and management of data in cloud computing. However, different indexing techniques were proposed by researchers with particular direction to big data in cloud computing. Approximate nearest neighbour (ANN) search is a fundamental for searching large-scale-image in application domains such as data mining, pattern recognition, machine learning and information retrieval [5], [6], [7], [8], [9], [10], [11]. Tree-based indexing techniques proposed by [12], [13], [14], [15], were proposed for similarity searching in low-dimensional space. A R-tree-based indexing technique was proposed in [16], to allow multi-dimensional data to be indexed in the cloud. Distributed B-tree is an indexing method designed to perform consistent and concurrent updates and at the same time permits high concurrency reading operations (Aguilera et al., 2008). Graph-based techniques [17], [18],[19], [20], [21], [22], have been proposed are known for their performance in terms of similarity searches and are mostly based on the idea that a neighbour of a neighbour is also a neighbour. Hash based indexing techniques [23], [24], [25], [26], [27], [28], [29], [31], [32], [33], [34], are known for their effectiveness in search accuracy and low memory requirements. Hashing based techniques have been used in application area like large-scale vision problems including image retrieval [35], [36], image search [37], object recognition [38], local descriptor compressing [39], fast multimedia search [40], image matching [41], and are efficient in search and similarity computation. Approximate similarity search [42], while the c^2[43], is used for maintaining index items in d-dimensional data.
The drawback of the afore mentioned schemes is that the performance of the system degrade as the code length increases which results to low speed, retrieval accuracy, and high search time.in this paper, we therefore propose a geometric similarity preserving embedding-based hashing to alleviate the aforementioned drawback posed thereby improving the search accuracy and memory cost.

The rest of this paper is organised as follows: the review of recent hashing techniques are given in section II, our proposed method is presented in detail in section III, reports based on our experimental findings are presented in section IV while section V conclude our work.

2. Related works.

Hashing based methods are known to be efficient in accuracy and memory requirements. The hashing based methods are categorised into data-independent and data-dependent. The data-dependent based hashing designed based on the distribution of data to overcome the inefficiency of data-independent hashing methods such as the LSH [50] that generates projections randomly which result to long code length and many hash tables. Learning based hashing techniques includes supervise, semi-supervise and unsupervised hashing techniques

For searching fingerprint in Hamming space, [44] proposed a theoretical framework for learning compact binary hash and an integrative technique to hash-based fingerprint indexing. The authors build their method on the existing Minutiae Cylinder Code (MCC). The characteristic of MCC are outlined and pointed out that it binary representation are bit correlated which makes it possible to represent minutiae feature in a more compact binary form. The theory of Markov Random Field (MRF) was used to model adjacent bit correlations in the MCC binary representation which makes it easier in learning the hash bit from a generalised linear model (GLM). To obtain an entropy for finding more descriptor bits and independent bit, equation. Experimental results based on 24-bit hash code show that the identification accuracy is high and faster search time

$$H(X_1, X_2, \dots, X_n) = \sum_{i=1}^{n} H(x_i) \quad (39)$$

A hashing technique that uses two hash codes of different length for stored images in the database was proposed in [45]. The compact hash code is used for the stored images in the database to reduce storage cost while the long hash code is used for the queries for searching accuracy. To retrieve images from the database, the Hamming distance of the long hash code is computed for the query and the cyclical concatenation of the compact hash code of the stored images for better precision-recall rate. The propose method is compared with Iterative Quantisation (ITQ), Iterative Quantisation with Random Fourier Feature (ITQRFF), Orthogonal K-mans (OKM), Asymmetric Hashing (AH), LSH and Shift-invariant Kernel based Locality Sensitive Hashing (SKLSH). This method gives a better performance than ITQFF because the asymmetric hashing approach of the ACH provides better precise location data of the query. The experimental results show that ACH yields the best precision with the code length of 64-bit than the existing methods. The drawback of ACH is that the technique cannot distinguish favourably between images of birds and planes dew to similar feature values used. The scheme depends on the semantic similarity preservation of the features used and the weighted Hamming distance increase the computational complexity of the scheme. [46], proposed a method to preserve the underlying geometric information among data. The authors explore the sparse reconstructive relationship of data to learn compact hash codes. Usually, it gets over fitting in measuring the empirical accuracy on labelled data as such information provided by each bit is utilised to obtain desired properties of hash codes. The information theoretic constraint is incorporated into the relaxed empirical fitness as a regularising term to obtain the objective function. [47], proposed a novel method for improving the performance of geometric hashing with respect to robustness towards occlusion and clutter. Features descriptors are used reduce the rate of false positives. To improve the performance of the geometric hashing, distinctive features given by the feature points are added to the representation and does not allow the cluttering of feature points to form bases pairs. These feature descriptors are used to discard features that can easily lead to data collision in the hash table. As a result, the number of bases that needs to be examining are being reduced in the recognition stage. By evaluating all over threshold bins in the Hash Table obtained with a particular basis, multiple objects instances are being handle into the target images. Experimental results show that the new method outperform GH techniques in terms of recall at 1-precision when the number of feature points used is 30 feature on the average per frame and 175 feature points on the average per frame. In the data dependent techniques, data points are of utmost consideration in designing hash functions. Binary code embedding methods provides high compression efficiency and fast similarity computation. [48], make use of a boosted geometric hashing to design an efficient indexing technique for finger-knuckle-print (FKP) images to reduce the searching cost. For effective design, the authors identify that the number of feature points are not the same, the number of extracted feature is huge, the number of features of two images of the same subject extracted at two different instances of time may not necessarily be the same, the images may be partially occluded, and a query image may be rotated and translated with respect to the corresponding image in the database. Features extraction algorithm are used to extract feature from each and every FKP image. The proposed method consist of two phases known as indexing and recognition. In the indexing step, to boost the geometric hashing for efficient indexing and recognition, the coordinate position of each feature is used to generate an index in the hash table while using the descriptor vector for recognition. Mean centering, the rotation of feature based on principal components and normalisation are used to handle translation, rotation and scaling effect present in the images. Extensive experiments were carried out to evaluate the performance of the proposed techniques as compared to other state-of-the-art-techniques based on Hit Rate, Penetration Rate, correction Recognition Rate, and Cumulative Match Characteristic (CMC) curve. The scheme is tested on publicly available PolyU finger-knuckle-print database. The results show that when SIFT and SURF were used to extract features from images, CRR achieve 96.36% and 99.69% respectively. Hit rate was achieved at 100% against penetration when the neighbouring are varied. The proposed technique is compared with the geometric hashing method where the features are inserted into the hash table once in the boosted geometric hashing technique and was found that the boosted geometric hashing outperform the conventional geometric hashing technique. [49], proposed a novel hashing algorithm for effective high dimensional nearest neighbour search. DSH uses k-means to roughly partition the data set into k-groups. Then for each pair of adjacent group, DSH generate one projection vector which can well split the two corresponding group. From the generated projections, DSH select the final ones according to the maximum entropy principle in order to maximise the information provided by each bit. Given n_i data points $X = [x_i, \dots, x_n] \in R^{il*n}$, is to find L hash functions to map a data point x to a L-bits hash code. In geometrical perspective, w_l defines a hyperplane. The points on the sides of the hyperplane have the opposite labels. This hash function, the hash bits of two points has the probability proportionate to the cosine of the angle between them. Basically, two points identify as $x_i, x_j \in R^d$. Based on this characteristic, the DSH has the probabilistic guarantees for retrieving data within $(1 + \mathrm{e})$ times the optimal similarity, and the query times that are sublinear with respect to n. DSH shows minimal improvement in performance as the code length increases because the geometric discriminative structure information of data is ignored and this result to a suboptimal performance of DSH. The DSH uses hyperplane-based hashing function to encode high-dimensional data and to partitioned data points into two sets and assigned two different binary codes (-1 or +1) depending on which set each point is assigned to. To minimise distortion in DSH, the centre point is used as the representative for each group. In large scale applications, it took long time for the k-means for the k-means to converge. After p iteration, the k-means is stopped. Also, the best group number is usually, is the large group with smaller error. But a large number of group could lead to high computational cost in the quantisation step. Now DSH uses the quantised result denoted by k groups $S_1, \dots, S_k$and the $i-th$ group has the centre μ_{ip} to guide in generating the projections. Here, the nearest neighbours matrix represented by W is given as in equation (52) which is used in defining the $r-$ adjacent groups that is $S_1$and S_k, if and only if $W_{ij}=1$. So the DSH now picks only those projections that can separate two adjacent groups properly.

Furthermore, for each $S_1$and S_k (pair groups), the DSH technique utilises the median plane between the centres of the adjacent groups as the hyperplane to separate points. Although hyperplane-based hashing function is used to obtain the median plane between the centres of adjacent groups, geometric discriminative data points should be bounded and mapped into binary hash codes by fining tighter closed region, is used to separate points. Again, a good binary code maximise information given by each bit. That is, maximum information is given by a binary bit that has a balanced partitioning of the data points. For this, each of the candidate's projection DSH computes its entropy. So for the entropy in DSH with regards to projections. DSH is evaluated on high-dimensional nearest neighbours search problems using three large scale real-world data sets conducted on experiments. Results from experiments show that scheme is suitable for high dimensional situations. With the codes of 48 bits and 96 bits, DSH has better precision-recall rate than state-of-the-art-techniques used as comparison algorithm. Using of hyperplane-based hashing function incurs more computational cost as $d+1$ hyperplanes are required to define a close region for a d-dimensional space. DSH generates more and more projections which are less important and becomes redundant. The redundancy of projections degrades the performance of DSH and thus does not scale well with huge databases. For this, an optimised algorithm is require to balance the trade-off between search accuracy and search time and still maintain memory cost.

3 Methodology

Here we present our proposed system and its operational principle. The proposed system is composed by four components that performs a specific function to achieve the set objectives. The objective of learning hashing-based methods is to use the mapping function $h(x)$ that projects m-dimensional real valued feature vector to n-dimensional binary hash codes and still preserve the similarity among the feature vector and the data set. Geometric Similarity Preserving embedding-based Hashing (Geo-SPEBH) can preserve the underlying discriminative geometric information among the data points. The system explores the magnitude structure of geometric features of data. Here the image feature are indexed from the quantised hashing results. The Geo-SPEBH uses hypersphere-based hashing function for computing the binary hash codes with a joint algorithm that optimised search accuracy and search time simultaneously.

We have sample of data points contained in a database, we will index the data to reduce storage cost, computational cost and optimise the search accuracy and time simultaneously. Here we represent the data point's samples as $x_1, x_2, x_3, \dots, x_N$, and the database is represented as X given below:

$X = \{x_1, x_2, x_3, \dots x_n, \dots, x_N\} \in R^{d \times N}$ denotes the data points contained in the database. Then we design our hash function that will map this data points to a k-bit binary hash code which we define by equation (1)

$$H(x) = \{h_1(x), \dots h_k(x)\} \in \{-1, 1 \qquad (1)$$

Where N is the number of samples of the data points, and k is the length of the binary hash code.

Figure 1 gives the conceptual framework of the proposed system. The working principle of the propose system is given in details with a detailed explanation of the responsibilities of each of the component that made up the model. This architecture incorporates the solutions to the identified problems in the various components that made up the proposed system.

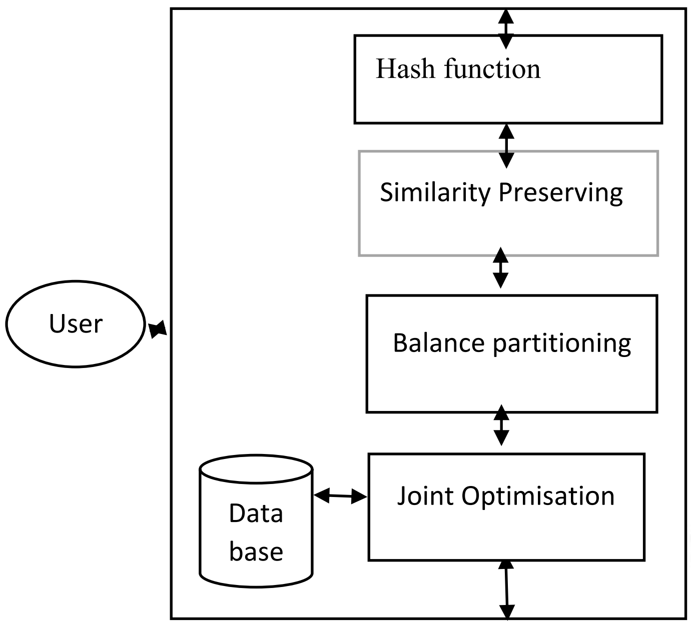

Figure 3.1. Conceptual framework of the proposed System.

Firstly, we use the hypersphere-based hashing function to project each sample data point into compact binary hash codes. The hash function depends on the distribution of data to generate hash table.

Secondly, we then use the geometric similarity preserving to preserve the similarity among the data points so that the original structure property of the data points are retained.

Thirdly, we then optimise the search accuracy and time simultaneously by minimising the Hamming distance and the mutual information to enhanced precision-recall.

3.1 Hash Function

This component of the propose system is responsible for the compression of the high-dimensional data into compact hash codes to minimise storage cost. The goal of designing the hash function is to preserve the similarity information of the original descriptor vector in a high-dimensional hamming space for better precision and recall. Here, the hash function of the propose system compresses the original descriptor of the stored images in the database to a low-dimensional k-bit compact binary hash code with high compression ratio for small storage cost. The low-dimensional binary embedding can preserve the original geometric coordinate structure information of the data in the database. We therefore construct our hash function using the geometric structure properties of the data based on the data distribution framework, and the hashing coding is depended on the data points. We use K hash functions based on hypersphere to preserve the similarity among data point samples. Our K hash functions ranges from 1 hash function to $K - th$ hash function. The constructed hash function is then use to generate binary hash codes $H\ (x_i\ ) = [h_1(x_i\ ), \ldots, h_k(x_i\ )]$ by compressing points in high-dimensional space into the binary hash codes of $H\ (x_i\ ) = \{-1, +1\}$. The samples in the original database of images which correspond to the non-negative entries are used to approximate the given data vector. To achieve this efficiently, a geometrical hashing function that utilises the hypersphere-based hashing function design in equation (1) is used to define a pivot in a D-dimensional vector space with a distance threshold. Our hashing function will show that the values represented by each of the geometrical hashing function $H\ (x_i\ )$ will then determine whether a data point say x, is within the range inside the hypersphere with the centre as c_i and it radius as w_i. The hash function is effective in that a higher number of region that are closed can be created using multiple hyperspheres, with distances between the points that are located in each of the region are bounded. To locate a nearest neighbours from a query point ANN search, closed region are formed with tight bounded distances. With this tighter regions, effective candidates for the nearest neighbours can be found within the range or region that is been indexed by the binary code of the query point.

We exploit the distribution of data using the geometric properties among data points to design in our hash function which takes a linear form as in equation (1) and equation (2) is where the hash codes are generated.

$$y_k = \frac{1}{2}\ (1 + h_k(x)) \qquad (2)$$

To overcome the low performance of hash function as the code length increases, we define independence of hashing functions to distribute data points evenly to the generated binary hash codes, and also informative set of projections as the number of the hash functions increases. We uses the hypersphere to define a geometrical hashing function. Here, a pivot $P_i \in R^D$ as a distance threshold $t_i \in R^+$ as the following equation (3). as $\{h_x\}_{k=1}^{K}$

$$h_i = \begin{cases} 0\ if\ (p_i, x)\ >\ t_i \\ 1\ if\ (p_i, x)\ \leq\ t_i \end{cases} \qquad (3)$$

Where $d(.\ ,\ .)$ denotes the Euclidean distance between two data points in R^D. To map similar feature vectors into the same hash bucket, every bit must have the opportunity of becoming one or zero. Here, similar bits are map into same bucket with high probability of having a 50% chance of becoming one (1) by defining independence of each bit. Equation (4) is use to balance the partitioning of data points for each bit.

$$p_r[h_i(x_i) = 1] = \frac{1}{2}, x\ \in X, 1 \leq i \leq t \qquad (4)$$

To achieve independence between two bits given that $x\ \in X$ and $1\ \leq i < j \leq t$

$$p_r\left[h_i(x) = 1, h_j(x) =\ 1\right] =\ p_r[h_i(x) = 1]\ .p_r\left[h_j(x) = 1\right] = \frac{1}{2} . \frac{1}{2} = \frac{1}{4} \qquad (5)$$

Our hash function will use a few entries indexed in the hash table to reconstruct the original data which ultimately depends on the distribution of the data points that utilises the intrinsic attributes of the data, that is, the underlying geometric structure properties of the data points in the data base. By using the few entries in the hash table, the original data is reconstructed and retrieved.

3.2 Geometric Similarity Preserving

This component of the propose system is responsible for preserving the similarities of two sample data points in the training data set in our propose system. Given a database X, two data samples X_i and

X_j contained in the training set of data. We extract the similarity between the two data samples as Q_{ij} from the similar geometric feature points of image data. Hashing methods require geometric coordinate properties for similarity preserving. We will then ensure that the data points that are similar usually have similar hash codes with small hamming distance.

The similarities among the sample data points detected using SIFT is then preserved as a similarity preserving term, and then we further seek a code that maps similar data points to similar binary hash codes known as similarity preserving. We then minimise the Hamming distance between similar data points and the corresponding similar binary hash codes. Our similarity preserving term, and Hamming distance minimisation between similar data points and it corresponding similar binary hash code are represented in equations (6) and (7) respectively. We sum the similarity preserving term as the summation of x_i samples of data points from 1 to N plus the summation of x_j corresponding similar binary hash code from 1 to N as in equation (6). We minimise the Hamming distance by taking the absolute values of the of the similarity term as in equation (7).

Hamming distance = taking the absolute (abs) values of Similarity term.

$$Q(y) = \sum_{i=1,\dots,N} x \sum_{i=1,\dots,N} x = \sum_{ij=1,\dots,N} x \qquad (6)$$

$$Q(y) = \sum_{i=1,\dots,N} \sum_{j=1,\dots,N} Q_{ij} ||Y_i - Y_j||^2 \qquad (7)$$

where Q_{ij} is the sample data that has similarity, $Q(y)$ is the similarity preserving term.

For efficient search accuracy with respect to similarity search, similar data points are map to similar binary hash codes for similarity preserving. This means that similar data points must have similar binary hash codes with small Hamming distance by minimisation. We use equation (7) to minimise the Hamming distance with respect to:

$$yi \in \{0,1\}^k \sum_i yi = 0 \qquad (8)$$

$$\frac{1}{n} \sum_i yiyi^T = I \qquad (9)$$

Where the constraint (8) requires each bit to fire 50% of the time, and the constraint (9) requires the bits to be uncorrelated. And, y is the set of all Y_i. Then from equation (7), samples with high similarity or with bigger similarity Q_{ij} will have similar binary hash codes with smaller Hamming distance $||Y_i - Y_j||^2$.

Process to Map similar data points to similar binary codes.

Input: extract feature points from images in the training dataset in the database using PCA.

Form a sphere around the extracted features using the geometric coordinates of feature points.

Perform PCA to the extracted feature points to make them invariant to rotation, scaling and transformation.

Extract similar feature points from sample data points $x_i and\ x_j$ using SIFT feature detector to detect similarities among data points Q_{ij}

Make Q_{ij} the similarity matrix among the sample data points

Seek a code to map similar data points to similarity binary hash codes to preserve similarity $Q(Y)$ in equation (6).

Minimise the Hamming distance between similar data points and their corresponding similar binary hash codes abs $Q(Y)$ in equation (7).

3.3 Balance Partitioning for independence.

To have uniform distribution of data points in hash bucket, we make each hash function independent of one another. That is the functionality of one hash function does not depend on the other one to function. Each hash function is depended on itself to distribute data points in an evenly manner. We therefore give each hash function the opportunity of becoming 0 or 1. This mean that for hash functions to be independent, each hash function should have the chance of being one or zero and the different binary hash codes are independent of each other as in equation (4) above.

We demonstrate the independence of our hashing functions as follows: As a typical scenario, we take the probability that an event say B_i be a hash function that is one (1). B_i is the event that $h_i(x) = 1$. We then define two events $B_i$and B_j to be independent if and only if the probability of $B_i = 1$ and the probability of $B_j = 1$is equivalent to the probability of $B_i = 1$ multiply by the probability of $B_j = 1$ as in equation (10). Here, similar bits are map into same bucket with high probability of having a 50% chance of becoming one (1) by defining independence of each bit. Any of equation (4) and (5) is use to balance the partitioning of data points for each bit.

$$p_r[h_i(x_i) = 1] = \frac{1}{2}, x \in X, 1 \le i \le t \qquad (4)$$

$$N_i = \sum_{i=1}^{2^M} N_i \qquad (11)$$

To achieve independence between two bits given that $x \in X$ and $1 \le i < j \le t$

$$p_r\left[h_i(x) = 1, h_j(x) = 1\right] = p_r[h_i(x) = 1] . p_r\left[h_j(x) = 1\right] = \frac{1}{2} . \frac{1}{2} = \frac{1}{4} \qquad (5)$$

$$Pr[Bi \cap Bj] = Pr[Bi] . \Pr[Bj] \quad (10)$$

$$p_r[h_i(x_i) = 1] = \frac{1}{2}, x \in X, 1 \le i \le t$$

The intersection is the equal chance of the code bit being 1.

We then incorporate the similarity preserving term with the minimised mutual information criterion together to simultaneously improve the search accuracy and search time. We insert the data points into each bucket as

$$Ni = \frac{N}{2^M} . \qquad (11)$$

Algorithm 1: Balanced Partitioning

```
1. Start
2. Let V = 2**M
3. Input:  N; M
4.     $for\ i = 1\ to\ V$
5. get $N(i)$
6. BP = N($i$) $**$ 2
7. $i = i + 1$
8.     $if\ i \le V$ goto step 5
```

```
9.        end if
10.     end for
11.  Print BP
12.  Stop
```

3.4 Joint Optimisation.

In this section, we integrate the similarity preserving term $Q(Y)$ for search accuracy and the minimum information criterion for the search time to form a single entity. To enable a high search accuracy with fast search time, the joint optimisation component of the propose system is formulated and is responsible for the simultaneous optimisation of the search accuracy and search time. A parametrisation of a linear function is perform for easy optimisation, and a relaxation is perform. To relax the similarity preserving term and the minimum information criterion, the binary constraint is being removed from the similarity preserving term and the minimisation of the minimum information criterion.

The joint optimisation is responsible for the computation of the hash bit that will be used for query and the identification of the bucket with the same hash bits with the query, and to also oversee the loading of data samples from the selected buckets into the memory. Here, we make the hashing function independent to distribute data points evenly or equally to different binary hash codes.

To minimise the time complexity, each bucket will contain equal number of samples to have a balanced buckets. To have equal number of samples in each bucket to balance the buckets, we use $N = \frac{N}{2^M}$ equation (11)

Here, we improve the search accuracy by minimising the Hamming distance between similar data points.

$Q(y) = \ sum\ of\ samples\ of\ x\ from\ i\ to\ N$ + $sum\ of\ samples\ of\ x\ from\ j\ to\ N$. Mathematically, this can be expressed as:

$$Q(y) \sum_{i=1,\ldots N} x + \sum_{j=1,\ldots N} x \qquad (12)$$

The similarity preserving term and the mutual information are incorporated together for simultaneous improvement in search accuracy and search time.

3.4.1 Similarity preserving term $Q(y)$

To improve the accuracy of searches in a database, we use the similarity preserving term which contains the similarity features among the data points, $Q(y)$, with a minimised Hamming distance in equation (7).

Algorithm 2: GSPEBH

```
1. Start
2. Input: the training dataset$X_i$, $i = 1,2,3, \ldots, N$, similarity matrix $W$ and $W = W_{ij}$; the number
   of required bits $K$ to map the full dataset as hash codes; BP; N; M;
3. Initialise: Sum = 0; Sim = 0; SimM = 0; BP = 0; V = 2**M; yi = 0; JointO = 0
4.              $for\ i = 1\ to\ c$
5.                   $for\ j = 1\ to\ c$
6.                   Get y($i$), y($j$), x($i, j$)
7. Sum = Sum + $(y(i) - y(j))$**2
```

```
8.                  j = j + 1
9.                 $if\ j \le c$ goto step 6
10.                 end if
11.                      $i = i + 1$
12.                      $if\ i \le c$ goto step 17
13.                      end if
14.                  end for
15.             end for
16.  Sim = Sum
17.               break;
18.                  $for\ i = 1\ to\ V$
19.                     get $N(i)$
20.                    BP = N($i$) $**$ 2
21.                    $i = i + 1$
22.                    $if\ i \le V$ goto step 40
23.                    end if
24.                 end for
25.  Print Sim, BP
26.  //Incorporating similarity preserving term and balanced partitioning//
27. JointO = Sim + BP
28. //computing $u_i$//
29. $T(a, b)$ = 0, swap = 0
30.  Get x
31.  Get b
32.               $for\ i = 1\ to\ a$
33.                    $for\ j = i + 1\ to\ b$
34.                       Get $T(i, j)$
35.                       j = j + 1
36.                      $if\ j \le b$ goto step 55
37.                      i = i + 1
38.                          $if\ i \le a$ goto step 55
39.                          end if
40.                       end if
41.                    end for
42.               end for
43.      $for\ i = 1\ to\ a$
44.          $for\ i = 1\ to\ b$
45. Swap = $T(i, j)$
46. $T(i, j) = T(j, i)$
47. $T(j, i) = swap$
48. $h(i)$ = $sign(T(j, i) * x(i) - b$
49.         $j = j + 1$
50.             $if\ j \le b$ goto step 66
51.         $i = i + 1$
52.             $if\ i \le a$ goto step 65
53.                             end if
54.                         end if
55.                     end for
56.                end for
57. for i = 1
```

```
58. Print h(i)
59. $i = i + 1$
60.         $if\ i \leq a$ goto step 78
61.          end if
62.  end for
63. //computing Binary hash code//
64. Get k
65.                 $for\ i = 1\ to\ k$
66.                     $for\ j = i + 1\ to\ k$
67.             $y(j) = (1 + h(j))/2$
68. Print $y(j)$// for any sample$x_i$in the database, compute its K hash bits $Y_i = H(Y_i) = sign\ (T^T x_i - b)$//
69.
70.
71.                 $j = j + 1$
72.                           $if\ j \leq k$ goto step 89
73.                           end if
74.                       end for
75.                  end for
76.  Print JointO
77.  Stop
```

Table 3.1 Parameters used

S/no	Symbol	Uses
1	$D(Y)$	A similarity preserving term
2	r	A representation for r −adjacent groups
3	α	A parameter that controls the numbers of groups
4	$h(x)$	A hash function
5	c_i	The centre of hypersphere
6	w_i	The radius of the hypersphere
7	X_i	represents data sample
8	X_j	represents data sample

4. Experiments

In this section, we will give a detail analysis of extensive experiments carried out on GIST 1M dataset to evaluate the performance of our algorithm and compare with various state-of-the-art techniques. The details of our experiment and results are presented below.

4.1 Use Dataset.

The dataset used for the evaluation of our technique is the **GIST-1M-960D** which contains one million GIST features. Each of the GIST features is represented by a 960-dim vectors.

Table 3.1

Dataset	Dimension	No. of base vectors	No. of query vectors	No. of learn vectors
GIST 1M	960	1,000,000	1,000	500,000

Our ground truth is defined by k-nearest neighbours computed be exhaustive, linear scan base on the Euclidean distance. For each query, the ground truth contain the vector numbers which begins at 0 of its k-nearest neighbours, ordered by increasing Euclidean distance.

The codes are made available publicly by authors of the compared methods used in this paper. Where the authors do not provides their codes, we implement the algorithm develop according to their paper and with the same optimisation as our designed codes in this paper.

1 .http://corpus-texmex.irisa.fr
2. http://www.vision.ee.ethz.ch/_zhuji/felib.html
3. http://www.cad.zju.edu.cn/home/dengcai/Data/NNSData.html
4. http://corpus-texmex.irisa.fr
5. http://www.cad.zju.edu.cn/home/dengcai/Data/DSH.html
6. http://www.unc.edu/_yunchao/itq.htm
7. http://www.cad.zju.edu.cn/home/dengcai/Data/DimensionReduction.html
8. http://www.cs.huji.ac.il/_yweiss/SpectralHashing/
9. http://www.ee.columbia.edu/_wliu/

10. http://www.cse.ohio-state.edu/_kulis/klsh/klsh.htm

4.2 Evaluation metrics.

The Geo-SPBH is compared with state-of-the-art-techniques to obtain the mean average precision based on parameter analysis, the precision-recall rate, and the search accuracy and search time trade-off. We implement our method to measure the performance of retrieval result on the mean average precision (MAP) using the **GIST 1M** dataset. The MAP measures the average of precision scores for each query. It is the area under the precision-recall curve for a set of queries. A large value of MAP will indicate a better performance.

4.3 Compared Techniques.

The algorithms used in the evaluation of the proposed system are the DSH, SHD, KLSH, LSH, AGH, SpH, SIKH, PCAH.

DSH: Density sensitive hashing is a semi-supervised based hashing techniques that combined the characteristics of data-independent and data-dependent hashing techniques. The projections are generated based on selective principles. This technique avoids the complete random selection and generates the projections based on selective principles. It is an extension of LSH.

LSH: Locality sensitive hashing is a hashing based technique that does not depend on the distribution of data for projection generation. LSH generates its projections randomly. The vectors use for projection generation are randomly sample from a p-stable distribution. It is an unsupervised technique that apply to change detection domain.

Spherical Hamming Distance: SHD is a hashing based technique that uses spherical Hamming distance.

Kernerlised Locality Sensitive Hashing: KLSH generalises the LSH to the kernel space.

Principal Component Analysis Hashing: PCAH is a classic indexing method for big data that utilises the core principal directions as projective vectors to obtain binary hash codes.

Spherical Hashing: SpH is a classic approach that quantised the values of analytical Eigen functions computed along the principal component analysis of the data. The PCA is conducted on original data.

Anchor Graph Hashing: AGH approaches is a classic technique design an anchor graph to accelerate the speed of the spectral analysis procedures.

Shift-Invariant Kernel Hashing: The SIKH is based on the random feature hashing for approximating the shift-invariant kernels.

Geo-SPEBH: This is our design method to be compared with the above mentioned algorithms.

4.4 Results

To generates discriminative binary hash codes that needs only small number bits to code a huge amount of data in a database to yield high search accuracy and an improved search time with less memory consumption. Develop a framework for a geometric similarity preserving embedding-based hashing for large scale image retrieval in cloud computing.

We use the GIST 1M dataset to implement our Geo-SPEBH algorithm and compared our result with state-of-the-art-techniques.

Table 3.2

	GIST 1M Mean Average Precision (%)					
	Code length (bits)					
METHODS	16	32	48	64	80	96
LSH	0.0675	0.0805	0.1095	0.1815	0.1525	0.1705
DSH	0.1200	0.1500	0.2000	0.2300	0.2407	0.2501
SHD	0.0245	0.0439	0.0685	0.0945	0.1000	O.1843
SpH	0.1913	0.1431	0.1518	0.1685	0.1683	0.1725
AGH	0.0840	0.0945	0.1345	0.1445	0.1525	0.1535

PCAH	0.0855	0.0995	0.1455	0.1650	1.1705	1.1742
SIKH	0.0501	0.0676	0.0718	0.0825	0.0998	0.1325
KLSH	0.0690	0.1000	0.1095	0.1102	0.1423	0.1486
GSPEBH	0.1245	0.1585	0.2545	0.3100	0.3505	0.3605

4.5 Discussion

The **GIST 1M** dataset is a dataset that consist of one million GIST features represented by 960 dimension vectors. The number of base vectors is 1,000, 000 while the query vectors 1,000, 500, 000 vectors are used for learning. This dataset is run with the old algorithm (DSH) with varied number of bits, 16, 34, 48, 64, 80, 96 to obtain the mean average precision (MAP) for each query. We select 1K data points as the queries and the remaining are used to form the gallery database. The point retrieved is seen as the true neighbour if it lies in the top 2 percentile points closest to the query. It is measured by the Euclidean distance in the original space. The data points in the database for every query are ranked according to their Hamming distances to the query.

It can be seen that the data-dependent based methods recorded a very low performance when the code length is short but achieved high performance when the code length is long. This methods are LSH, SIKH, and KLSH. The data-dependent techniques, PCAH, SpH, SHD, AGH and semi-supervised technique recorded a high MAP performance when the code length is short but drops when the code length increases. From table 3.2 above, it can be seen that our proposed method outperform the compared methods as it recorded high MAP when the code length is short and still maintain performance when the code length increases, and the memory cost is low compared to the base-line methods on the GIST 1M dataset for all code lengths. The low memory cost recoded by our proposed algorithm indicate that it can handle large amount of data (huge database). Table 3.2 gives the MAP results for the GIST dataset for all the compared methods. Given 0.3 MAP obtain from the results above, our Geo-SPEBH requires 64 bits to encode each image in the sample dataset. On the other hand, the compared methods requires more than 64 bits up to 80 bits to encode each image in the database.

5 Conclusion

In this paper, we have proposed a geometric similarity preserving embedding-based hashing algorithm to address the drawback of the existing methods whose performance degrade as the code length increases. The high MAP achieved indicate that the proposed method designed for improving the search accuracy by using the similarity preserving term and minimising the Hamming distance. Experiment conducted using GIST 1M demonstrate that the proposed method achieve better performance compared to state-of-the-art methods based on MAP.

References

[1] Thilkanathan, Danan, S. C., Surya, N., Rafael, C., & Leila, A. (2014). A platform for monitoring and sharing of generic health data in the cloud. *Future generation computer system, 35*, 102-113. Retrieved April 9, 2017

[2] A. Labrinidis H. Jagadish. (2012) Challenges and opportunities with big data. Proceedings of the VLDB Endowment, 5(12): 2032–2033

[3] J. Wang, W. Liu, S. Kumar, S.Chang,. (2015). Learning to hash for indexing Bid data-A survey. Proc. Of IEEE. Pp. 1-22.

[4] J. Lu, Y. Lu, Cong G. Reverse spatial and textual K nearest neighbour search. In: Proceedings of the 2011 International Conference on Management of Data. 2011, 349–360.

[5] S. J. Beis and G. D. Lowe. 1997. Shape Indexing Using approximate Nearest-Neighbour Search in High-Dimensional Spaces. In 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97). 1000–1006.
[6] H. Ben and D. Tom. 2016. FANNG: Fast Approximate Nearest Neighbour Graphs. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. 5713–5722.
[7] K. P. Bennett, U. Fayyad, and D. Geiger. 1999. Density-based indexing for approximate nearest-neighbor queries. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 233–243.
[8] L. Chen, M. T. Özsu, and V. Oria. 2005. Robust and fast similarity search for moving object trajectories. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM, 491–502.
[9] A. P. de Vries, N. Mamoulis, N. Nes, and M. Kersten. 2002. Efficient k-NN search on vertically decomposed data. In Proceedings of the 2002 ACM SIGMOD international conference on Management of data. ACM, 322–333.
[10] G. Teodoro, E. Valle, N. Mariano, R. Torres, W. Meira, and J. H. Saltz. 2014. Approximate similarity search for online multimedia services on distributed CPU–GPU platforms. The VLDB Journal 23, 3 (2014), 427–448.
[11] Y. Zheng, Q. Guo, A. KH Tung, and S. Wu. 2016. Lazylsh: Approximate nearest neighbour search for multiple distance functions with a single index. In Proceedings of the 2016 International Conference on Management of Data. ACM, 2023–2037.
[12] J. L. Bentley, "Multidimensional binary search trees used for associative searching," Communications of the ACM, vol. 18, pp.509–517, September 1975.
[13] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," ACM TOMS, vol. 3, no. 3, pp. 209–226, 1977.
[14] A. Guttman, "R-trees: a dynamic index structure for spatial searching," SIGMOD Rec., vol. 14, no. 2, pp. 47–57, Jun. 1984.
[15] M. Muja and D. G. Lowe, "Fast approximate nearest neighbours with automatic algorithm configuration," in VISAPP, 2009, pp.331–340.
[16] J. Wang, Wu, S., Gao, H., Li, J., & Ooi, B. C. (2010). Indexing Multidimensional Data in a Cloud System. ACM SIGMOD International conference on management of data, 591-602.

[17] Yubao Wu, Ruoming Jin, and Xiang Zhang. 2014. Fast and unified local search for random walk based k-nearest-neighbor query in large graphs. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 1139–1150.
[18] Sunil Arya and David M Mount. 1993. Approximate Nearest Neighbor Queries in Fixed Dimensions.. In SODA, Vol. 93. 271–280.
[19] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast Approximate Nearest-Neighbor Search with k-Nearest Neighbor Graph.. In IJCAI 2011, Proceedings of the International Joint Conference on Artificial Intelligence, Vol. 22. 1312–1317.
[20] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems 45 (2014), 61–68.
[21] Godfried T Toussaint. 1980. The relative neighbourhood graph of a finite planar set. Pattern recognition 12, 4 (1980), 261–268.
[22] Zhongming Jin, Debing Zhang, Yao Hu, Shiding Lin, Deng Cai, and Xiaofei. He. 2014. Fast and Accurate Hashing Via Iterative Nearest Neighbors Expansion. IEEE transactions on cybernetics 44, 11 (2014), 2167–2177.
[23] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Localitysensitive
hashing scheme based on p-stable distributions," in SoCG, 2004.
[24] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in NIPS, 2008.
[25] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting," in BMVC, 2008.
[26] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in CVPR, 2008.
[27] M. Raginsky and S. Lazebnik, "Locality sensitive binary codes from shift-invariant kernels," in NIPS, 2009.
[28] J. Wang, S. Kumar, and S.-F. Chang, "Sequential projection learning for hashing with compact codes," in ICML, 2010.
[29] L. Pauleve, H. J´egou, and L. Amsaleg, "Locality sensitive hashing: a comparison of hash function types and querying mechanisms," Pattern Recognition Letters, 2010.
[30] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in CVPR, 2010.

[31] R.-S. Lin, D. Ross, and J. Yangik, "Spec hashing: Similarity preserving algorithm for entropy-based coding," in CVPR,2010.
[32] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in ICML, 2011.
[33] Y. Gong and S. Lazebnik, "Iterative quantization: a procrustean approach to learning binary codes," in CVPR, 2011.
[34] A. Joly and O. Buisson, "Random maximum margin hashing," in CVPR, 2011.
[35] Prateek Jain, B. K. (2008, 6). Fast image search for learned metrics. In proceeding of the IEEE conference on computer vision and pattern recognition.

[36] Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., & Yu, N. (2011). Complementary hashing for approximate nearest neighbor search. In Proc. ICCV.

[37] Kulis, B., Jain, P., & Grauman, K. (2009). Fast similarity search for learned metrics. TPAMI, 31(12), 2143–2157.

[38] Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 million tiny images: a large dataset for non-parametric object and scene recognition. TPAMI, 30(11), 1958–1970,.

[39] Strecha, c, A. M., M, M. B., & P, F. (2012). Ldahash: Improved matching with smaller descriptors. TPAMI, 34(1), 66-76.

[40] Zhu, X., Huang, Z., Cheng, H., Cui, J., & Shen, H. T. (2013). Sparse hashing for fast multimedia search. ACM Transaction on information system, 3(2), 1-24.

[41] Avidan, S., & Korman, S. (2011). Coherency sensitive hashing. In Proceedings of ICCV.

[42] Datar, M., Immorlica, N., Indyk, P., & Mirrokni, P. (2004). Locality sensitive hashing scheme based on p-stable distributions. In Proceedings of the Symposium on Computational Geometry, 253–262.

[43] Zhou, A. (2005). c^2: a new overlay network based on can and chord. international journal of high performance computing network, 3(4), 248-261.

[44] Wang, Y., Lipeng, W., Yiu-Ming, C., & pong, C. Y. (2015, August). Learning Compact Binary Codes for Hash-Based Fingrprint Inexing. *IEEE Transation on Information Forensic and security, 10*(8), 1603-1614.

[45] Yueming, L., Wing, W. Y., Ziqian, Z., Daneil, S. Y., & Patrick, P. K. (2015, August). Asymetric Cyclcial Hashing for Large-Scale-Image Retrieval. *IEEE Transaction on Multimedia, 17*(8), 1225-1235.

[46] Ye, R., & Xuelong, L. (2016, March). Compact Structure Hashing Via Sparse and Similarity Embedding. *IEEE Transactions on Cybernetics, 46*(3), 718-728.

[47] Tombari, F., & Luigi, D. S. (2011). Improving geometric hashing by means of features descriptors. International Conference on Computer Vision Theory and Applications, 419-425.

[48] Jayaraman, U., Aman, K., & Phalguni, G. (2014). Boosted geometric hashing based indexing technique for finger-knuckle-print database. *Information sciences, 275*, 30-44.

[49] Jin Z, L. C., Lin, Y., & Cai, D. (2014, august). Density Sensitive Hashing. *IEEE transactions on Cybernetics, 44*(8), 1362-1371.

[50] M. S. Charikar, "Similarity estimation techniques from rounding algo- rithms," in *Proc. Annu. ACM Symp. Theory Comput.*, 2002, pp. 380–388.