Received: June 7, 2021. Revised: August 13, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Minimum Search-time Algorithm for Image Retrieval in Cloud Computing

Abubakar Usman Othman[1]* **Aisha Yahaya Umar[2]** **Maryam Maishanu[1]**
Hauwa Abubakar[3] **Boukari Souley[1]** **Abdulsalam Ya'u Gital[1]**

[1]*Faculty of Science, Department of Mathematical Sciences, Abubakar Tafawa Balewa University Bauchi, Nigeria*
[2]*Department of Computer Science, Gombe State University, Nigeria*
[3]*Department of Computer Science, Umar Suleiman College of Education, Gashua, Nigeria*
* Corresponding author's Email: othman80s@yahoo.com

Abstract: Finding approximate nearest neighbour (ANN) is essential in huge database for efficient similarity search to return the nearest neighbour of a given query. Many hashing algorithms have been designed to improve retrieval accuracy and storage requirements of data in a large-scale database through long code word which increases the time complexity in loading data into memory, but do not consider the search time which is an important parameter in the field of information retrieval and pattern recognition. To address the aforementioned problem, this research therefore proposes an improved search time algorithm for improving the retrieval time of data from a database in cloud computing environment by optimising both the search accuracy and search time simultaneously. We improved the minimum search time by the use of balance partitioning algorithm for the even distribution of data points into hash buckets to minimise search time, and similarity preserving algorithm for search accuracy were designed for fast and accurate retrieval of data in a database. An extensive experiment conducted on a cloud simulator and the result obtained when the code length is 8, 96 bits, the retrieval time for the proposed system is 0.030sec, 0.260sec, and that of Density Sensitive Hashing is 0.040, 0.400sec. Therefore, the retrieval difference is 0.010sec and 0.140sec. Also, the result obtained for the rest of the code lengths of 16, 32 and 64 show that the improved minimum search time algorithm outperforms the compared techniques in terms of the velocity of big data retrieval.

Keywords: Balance partitioning, CloudSim, Cloud computing, Data, Hashing, Information retrieval, Time.

1. Introduction

Cloud computing is a web-based application that provides a shared pool of resources. The advance in mobile technology have allowed mobile devices such as smartphones and tablets to be used in a variety of different applications [1]. The availability of internet such as with the use of the wide spread broadband Internet access [2, 3], coupled with these hand held devices (mobile devices), resulted to the easy collection of digital information in form of structured and unstructured [4] data, had contributed to the availability of large volumes of data known as big data. Tremendous amount of data are generated every day in Manufacturing, Business, Financial Services, Science sectors and human personal lives. Adequate and proper processing of these data is required to open new discoveries and knowledge concerning markets, societies and human environment [5-7]. As unstructured data contributed to the availability of big data, they need to be structuralised for its effective understanding and processing through some optimised techniques used for extracting information. These information extracting techniques have been vastly used to extract meaningful information from raw or unstructured data [8]. Data has greatly changed and influenced researches in sciences. The Sloan digital sky survey is used by astronomers nowadays as a pool of resources which serve as a database [9, 10]. Biological data and experimental data are stored in a public storage facility and databases are created such that other biologists and scientists can make use of these generated biological and scientific data.

Many hashing based indexing techniques were also proposed to overcome the growing volume and search time for effective retrieval and management of big data. Hashing based indexing techniques can be grouped into two. The Data-independent hashing techniques generates its projection randomly to preserve the pairwise distances for data points, and the Data-dependent hashing techniques that mapped similar data pints to similar binary hash codes. The data-independent hashing techniques uses many random vectors for projection generation and these techniques includes [11, 12]. Data-dependent methods make complete use of the structure of data to generate short-code-word to achieve high precision and low memory cost and they includes [13-22], and improved data-dependent hashing techniques [23-26], were also proposed to reduce storage cost and to improve the precision rate of image retrieval as well as in a large database but incur high search time due to the time complexity involve in loading queries into memory. The time complexity slow the speed of data retrieval. The paramount decision when choosing and using supervised hashing techniques is based on the choice of similarity encoding approach. [14], proposed a novel graph-based hashing algorithm that automatically discovers the neighbourhood structure inherent in the data to learn compact hash codes. To make such an approach computationally realisable, the authors made use of Anchor Graphs to obtain tractable low-rank adjacency matrices. The use of a hierarchical threshold learning procedure in which each eigenfunction yields multiple bits, leading to higher search accuracy. Despite the gain, the anchor graph hashing algorithm, the use of long hash odes increase memory consumption and search time. [17], proposed a Locality-Sensitive Binary Codes that is similar to spectral hashing computationally, but is derived from completely different considerations, is amenable to full theoretical analysis, and shows better practical behaviour as a function of code size. They start with a low-dimensional mapping of the original data that is guaranteed to preserve the value of a shift-invariant kernel, specifically, the random Fourier features of Rahimi and Recht [12]. [27], proposed a supervised FastHash algorithm with a two-step learning strategy that uses binary code inference and followed it by binary classification that uses an ensemble of decision trees. This method achieved high performance with respect to search accuracy but failed to consider the minimum search time in their work.

The drawback of these schemes is that their performance for a better precision needs long hash codes which consumes large storage space. Also, the long hash codes slow the speed of retrieval since the search time depends on the number of data points present in a hash table. In addition, high search time is incurred due to the time complexity involve in loading queries into memory. The time complexity slow the speed of data retrieval. Researchers are often faced with the difficulties of designing a suitable research platform when carrying out research in cloud computing [28]. Also, the cost of setting up a cloud for the benefit of conducting research by scholars on live cloud is highly exorbitant [29]. In lieu of the above drawbacks of the existing hashing techniques, this paper therefore, proposes an improved balance partitioning algorithm to improve the minimum search time and similarity preserving algorithm for search accuracy by minimising the time complexity involve in loading queries into memory. Equal number of data points are distributed into hash buckets to improve search time. To outline the main contribution of this paper, advantages of the proposed improved search time algorithm is as follows:

i. Search time of data depends on the number of data points present in any selected hash table. For these, we designed an improved minimum search time algorithm to evenly distribute data points to each hash table for high speed of information from a database.
ii. We provided an algorithm for computing the minimum search time difference in algorithm 2.

The rest of the paper is organised as follows: related works are presented in Section 2, methodology of the improved search time and similarity preserving algorithms were presented in Section 3. Reports based on our experimental findings and discussions are presented in Section 4. Conclusions of our work are summarised in Section 5.

2. Related works

2.1 Indexing techniques

[30], proposed a novel robust discrete code modelling algorithm that learns high quality discrete codes and hash functions by supressing the influence of unreliable binary codes and potentially noisily-labelled samples. The robust discrete code modelling algorithm uses the $t_{2,p}$ norm to induce sample-wise sparsity and simultaneously perform selection of code and identification of noisy samples. [31], proposed an image authentication algorithm based on robust image hashing with geometric correction that eliminates the influence of geometric transformation, composite rotation-scaling-translation. Local features and global features are incorporated to construct hash functions. The local features were extracted from the salient regions using the Markov absorption probabilities. The global features used were the

statistical feature distance. The receiver operation characteristics show that the proposed image authentication algorithm shows superiority over some state-of-the-art techniques. [32], proposed a novel supervised discrete discriminant hash codes and hashing function simultaneously. The algorithm learned a robust similarity metric so as to maximise the similarity of the same class discrete hash codes and minimise the similarity of the different class discrete hash codes simultaneously in order to learn discrete hash codes to be optimal for classification. The discriminant information of the training data can then be incorporated into the learning framework.

Image retrieval in the field of multimedia application have recorded a considerable percentage of successes using the Deep hashing indexing approaches for similarity search. [33], proposed a binary generative adversarial networks for image retrieval to address the challenges of generating binary codes directly without relaxation, and equipping the binary representation with the ability of accurate image retrieval. Binary generative adversarial network (BGAN) were used to embed images to binary codes through the unsupervised approach. A new sign-activation strategy and a loss function was also proposed to steer the process of learning that consists of new models for adversarial loss, a content loss, and a neighbourhood structure loss. To extract features for the encoder, the author uses a structure of 5 groups of convolution layers and 5 maximum convolution pooling layers for the hashing, a binary code is learned directly by converting the L-dimensional representation z learned from the previously connected layer that is continuous in nature to the binary hash code b with respect to either $+1\ or-1$. [34], proposed a hashing-based-estimator for kernel density in high dimensions. Given a set of data P and a kernel function that returns approximation to the kernel density of a query point in sub-linear time. They introduce a class of unbiased estimators for bounding the variance of such estimators. The resultant estimators give rise to efficient data structures for estimating the kernel density in high dimensions for different commonly used kernels. [35], propose a large graph hashing with spectral rotation scheme by imposing spectral rotation techniques to the spectral hashing objective. The authors minimise the Euclidean distance in the modified solution to obtain the binary codes for index generation. This will result to semantical correlation with manifold where codes constraint is held. [36], proposed a novel hypersphere-based hashing function to map more spatial coherent data points into a binary hash code with a new binary code distance function suitable to the hypersphere-based coding scheme. [37], proposed a novel hashing algorithm for effective high dimensional nearest neighbour search. Density Sensitive Hashing (DSH) uses k-means to roughly partition the data set into k-groups. Then for each pair of adjacent groups, Density Sensitive Hashing generates one projection vector which can well split the two corresponding. 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 $X = [x_i, \ldots, x_n] \in R^{i*n}$, is to find L hash functions to map a data point x to a L-bits hash code.

$$H(x) = [h_1(x),\ h_2(x), \ldots, h_L(x)] \qquad (1)$$

Where $h_1(x) \in \{0, 1\}$ is the $l-th$ hash function.

Despite the gains in [37], there is 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 Density Sensitive Hashing (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. The proposed Geo-SPEBH (Geometric similarity preserving embedding-based hashing), hypersphere-based hashing function are used to encode proximity regions in high-dimensional spaces. The use of hypersphere improves the performance of search accuracy and time as the code length increases. 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.

Despite their successes, the above mentioned techniques are limited due to the longer time it take to search and retrieve data, and long hash codes consumes large memory thereby increasing the storage cost while the short hash codes gives unsatisfactory performance in terms of retrieval time. For this, an optimised algorithm is require to improve the search time and still maintain low memory cost.

The Geo-SPEBH Hashing aims at overcoming the drawback of data-independent based hashing methods and data-dependent based hash methods with respect to minimum search time, computational cost and memory cost. To guarantee the performance will increase as the code length increases, Geometric similarity preserving embedding-based hashing adopt the framework as Density Sensitive Hashing (density sensitive hashing). While Density Sensitive Hashing uses the geometric structure of data to guide the projections (hash tables) selection, Geo-SPEBH makes use of the geometric properties of principal component of features, which are confirmed to be very discriminative, and ensure that fewer features are inserted into the hash table. By using fewer features into the hash table and the balance partitioning algorithm that distributes data points uniformly into hash buckets, the minimum search time, the computational cost and memory cost will be greatly reduced.

3. Proposed method

Here we present our proposed system and its operational principle. The proposed system is composed by four components that performed each 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. The proposed method 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 features are indexed from the quantised hashing results. The Geometric similarity preserving embedding-based hashing uses hypersphere-based hashing function for computing the binary hash codes with a joint algorithm that optimise search accuracy and search time simultaneously. Samples of data points are contained in a database which will be indexed to reduce storage cost, computational cost and optimise the search accuracy and time simultaneously. Here we represent the data points' 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. Where X is the database and $R^{d \times N}$ represents the dimensional space of size N. Then we design our hash function that will map these data points to a k-bit binary hash code by Eq. (2)

$$H(x) = \{h_1(x), \dots h_k(x)\} \in \{-1, 1\}^k \qquad (2)$$

Where k is the length of the binary hash code and x. Represents the data points.

3.1 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 Eq. (11), [38]. This component of the proposed 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. Extracting the similarity between the two data samples as Q_{ij} from similar geometric feature points of image data is done. Hashing methods require geometric coordinate properties for similarity preserving. Next, the data points that are similar are ensured to have similar binary 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. The Hamming distance is then minimised between similar data points and the corresponding similar binary hash codes. The similarity preserving term, and Hamming distance minimisation between similar data points and it corresponding similar binary hash code are represented in Eqs. (3) and (4) 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 Eq. (3). Hamming distance is minimised by taking the absolute values of the of the similarity term as in Eq. (4) [38].

Hamming distance = taking the absolute (abs) values of Similarity term by Eqs. (3) and (4).

$$Q(y) = \sum_{i=1,\dots,N} x \sum_{i=1,\dots,N} x = \sum_{ij=1,\dots,N} x \qquad (3)$$

$$QH(y) = \sum_{i=1,\dots,N} \sum_{j=1,\dots,N} Q_{ij} ||Y_i - Y_j||^2 \qquad (4)$$

Where Q_{ij} is the sample data that has similarity, $Q(y)$ is the similarity preserving term and $QH(y)$ is the absolute value of the similarity term $Q(y)$.

For efficient search accuracy with respect to similarity search, similar data points are mapped 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 through Eqs. (5) and (6).

$$\sum_i yi = 0 \qquad (5)$$

$$\frac{1}{n} \sum_i yiyj^T = I \qquad (6)$$

Where the constraints Eq. (5) require each bit to fire 50% of the time, and the constraint Eq. (6) requires the bits to be uncorrelated. And, y is the set of all Y_i. Then from Eq. (4), samples with high similarity or with bigger similarity Q_{ij} will have similar binary hash codes with smaller Hamming distance $||Y_i - Y_j||^2$. Y_i and Y_j are the similar binary hash codes.

3.2 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. This is because each hash function is depended on itself to distribute data points in an evenly manner to different hash codes. Therefore, each hash function is given the opportunity of becoming 0 or 1 since binary digits are represented by zeros (0's) and ones (1's). This means 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 Eq. (8) above. Independence of hash functions is demonstrated in a scenario as follows: As a typical scenario, the probability that an event say B_i be a hash function that is one (1). B_i is the event that $h_i(x) = 1$. Then define two events $B_i$and B_j, next 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 Eq. (10). Here, similar bits are mapped into same bucket with high probability of having equal chance of becoming one (1) by defining independence of each bit. Any of Eqs. (7) and (8) is used to balance the partitioning of data points for each bit.

$$p_r[h_i(x_i) = 1] = \frac{1}{2}, x \in X, 1 \leq i \leq t \qquad (7)$$

$$N_i = \sum_{i=1}^{2^M} N_i \qquad (8)$$

Where N_i is the number of training samples in the ith bucket and M is the number of buckets,. p_r is probability. To achieve independence between two bits given that $x \in X$ and $1 \leq i < j \leq t$ where i and j are the ith and jth data points, and t is the threshold, hash functions are design to be independent Eqs. (9) and (10), and the data points are distributed equally to each hash bucket.

$$p_r\left[h_i(x) = 1, h_j(x) = 1\right] = p_r[h_i(x) = 1] \,.\, p_r\left[h_j(x) = 1\right] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \qquad (9)$$

$$Pr[Bi \cap Bj] = Pr[Bi] \,.\, \Pr[Bj] \qquad (10)$$
$$p_r[h_i(x_i) = 1] = \frac{1}{2}, x \in X, 1 \leq i \leq t$$

p_r is probability. The intersection is the equal chance of the code bit being a binary hash code 1.

The next is to incorporate the similarity preserving term with the balance partitioning components or terms together to simultaneously improve the search accuracy and search time. We insert the data points into each bucket in Eq. (8)

$$Ni = \frac{N}{2^M} \qquad (8)$$

3.3 Optimisation of search accuracy and minimum time

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 proposed system is formulated and is responsible for the simultaneous optimisation of the search accuracy and search time. A parameterisation of a linear function is performed for easy optimisation, and a relaxation is performed. 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, the hash function independent is made to be 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. This is done to minimise the search time. To have equal number of samples in each bucket to balance the buckets, $N = \frac{N}{2^M}$ [38], Eq. (8).

Here, the search accuracy is improved 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 Eq. (11):

$$Q(y) = \sum_{i=1,\ldots N} x + \sum_{j=1,\ldots N} x = \sum_{ij=1,\ldots,N} x \qquad (11)$$

The similarity preserving term and the balance partitioning are incorporated together for simultaneous improvement in search accuracy and minimum search time, [38]. The minimum search time (MST) is the minimum time taken from when a query is sent to when relevant data are retrieved from a database. Here we represent the time a query is sent as lambda λ that is the start time, while the finished time is represented as miu μ that is the time when data are retrieved. Therefore, minimum search time is calculated as the finished time minus the start time as given in Eq. (12) below.

$$MST = \mu - \lambda \tag{12}$$

Where MST is the minimum search time taken for data to be retrieved, λ is the time when a query was sent and μ is the time when a data were retrieved from a database.

Algorithm 1. Balance Partitioning

```
1.  Start
2.  Let V = 2**M
3.  Input:  N; M//N is the number of training
    sample in the $ith$ bucket//
4.  //M is the number of buckets//
5.  $for\ i = 1\ to\ V//V$ is the memory location for
    2**$M$//
6.  get $N(i)$
7.  BP = N($i$) $**$ 2
8.  $i = i + 1$
9.      $if\ i \le V$ goto step 6
10.     end if
11.    end for
12.  Print BP//output balance partitioning//
13.  Stop
```

Algorithm 2. Minimum Search Time

```
1.  Start
2.  //Algorithm to calculate minimum search
    time//
3.  Input: λ; μ// λ is the start time for search//
4.  //μ is the finished search time//
5.  $MST$ = μ - λ
6.  //$MST$ is the time taken for the minimum
    search//
7.  Print $MST$//output minimum search time.
8.  stop
```

Algorithm 3. Improved search accuracy and time

```
1.  Start
2.  Input: the training dataset$X_i$, $i = 1,2,3,\dots,N$,
    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//jointO is the
    memory location for joint optimisation
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.  $h(i)\ =\ sign(T(j,i) * x(i) - b$ //T is the
     projection matrix of $d \times M$ and $b$ is a vector//
49.           $j = j + 1$
50.               $if\ j \le b$ goto step 45
51.          $i = i + 1$
52.               $if\ i \le a$ goto step 44
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.         $if\ i \le a$ goto step 78
61.         end if
62. end for
63. Stop
```

4. Experimental setup

4.1 Performance metrics

The metric used to compare the proposed method is time. The Geo-SPEBH will be compared with existing techniques to obtain the minimum search time. We implement our method to measure the performance of retrieval result on the minimum time taken to send a query to when the data is retrieved from the database using the **SIFT 1B** dataset [39]. It is calculated using the time of the system used for the conduct of the experiment.

4.2 Comparison competitors

The technique used in validating the performance of the improve algorithm is simulation. The evaluation method used is time. The competitor algorithms used in the comparison are as follows:

1. Robust Discrete Code Modelling (RDCM) learns high quality discrete codes and hash functions [30].
2. Robust Geometric Correction (RGC) eliminates the influence of geometric transformation, composite rotation-scaling-translation [31].
3. Discrete Discriminant Hashing (DDH) The algorithm learned a robust similarity metric so as to maximise the similarity of the same class discrete hash codes and minimise the similarity of the different class discrete hash codes simultaneously [32].
4. Binary Generative Adversarial Networks (BGAN) use binary generative adversarial network embed images to binary codes through the unsupervised approach [33].
5. Large Graph Hashing (LGH) minimised the Euclidean distance in the modified solution to obtain the binary codes for index generation [35].
6. Spectral Hashing (SpH) is a classic approach that quantised the values of analytical Eigen functions computed along the principal component analysis of the data [36].
7. Density Sensitive Hashing (DSH)**:** is a semi-supervised based hashing techniques that combined the characteristics of data-independent and data-dependent hashing techniques. [37].
8. **Geo-SPEBH:** This algorithm used the geometric similarities of data points and also distributes the data points evenly into hash buckets [15].

4.3 System requirements and tools

All the experiments were conducted and run on a

Table 1. SIFT 1Billion data set use in implementing the existing system

Dataset	Dimension	No. of base vectors	No. of query vectors	No. of learn vectors
SIFT 1B	128	1,000,000,000	10,000	100,000,000

Table 2. Simulation results for the proposed and existing methods

	Time (sec) Code length (bits)					
METHODS	8	16	32	48	64	96
RDCM [30]	0.053	0.108	0.300	0.390	0.490	0.630
RGC [31]	0.053	0.080	0.248	0.310	0.412	0.522
DDH [32]	0.041	0.189	0.249	0.336	0.446	0.568
BGAN [33]	0.060	0.192	0.362	0.591	0.766	0.880
LGH [35]	0.040	0.080	0.160	0.280	0.330	0.420
SpH [36]	0.050	0.160	0.320	0.480	0.630	0.798
DSH [37]	0.040	0.080	0.160	0.240	0.320	0.400
Geo-SPEBH [15]	0.030	0.060	0.110	0.170	0.200	0.260

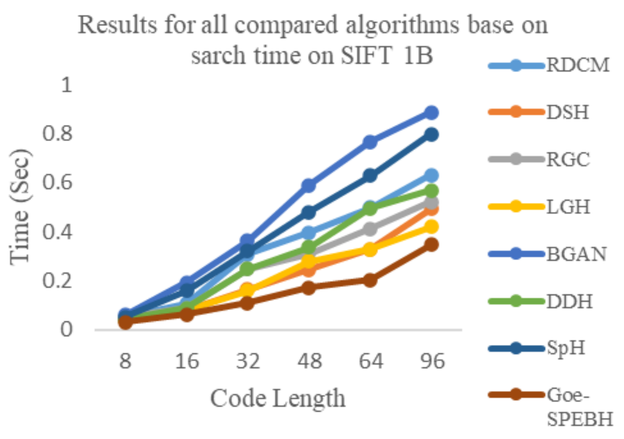

Figure. 1 Search time for random-based algorithms on SIFT 1B dataset

3.40 GHz CPU with four cores and 16 G RAM, in a Java software tool built on CloudSim for experimentation, simulation and implementation. The CloudSim is configured with 1 data centre on 100 cloudlets with the capacity of accepting input and output size of 300 each and length of 5000. The bandwidth requirement is 1GB network bandwidth.

4.4 Simulation results

SIFT 1B dataset from simulation results carried out on the existing algorithm implemented on the proposed system Geometric similarity preserving embedding-based hashing algorithm compared the result with some state-of-the-art-existing techniques. The results obtained are listed in Table 2 and Fig. 1.

The SIFT BM [39] dataset is a dataset that consist of one million SIFT features represented by 128 dimension vectors. The number of base vectors is 1,000,000,000 while the query vectors 10, 000, 100,000,000 vectors are used for learning. This dataset is run with all the algorithms with varied number of bits, 8, 16, 34, 48, 64, 96 to obtain the velocity of data retrieval for each query sent. We select 1K data points as the queries and the remaining are used to form the gallery database. It can be shown in Table 2 and Fig. 1 that the Geometric similarity preserving embedding-based hashing algorithm takes less time to retrieved data from a database because of the use of hyper-sphere and the even distribution of data points to each hash buckets. When the code length is 8, Geometric similarity preserving embedding-based hashing algorithm has a retrieval time of 0.03 sec. and 0.053, 0.053, 0.041, 0.060, 0.040, 0.050, and 0.040 sec. respectively for Robust Discrete Code Modelling [30], Robust Geometric Correction [31], Discrete Discriminant Hashing [32], Binary Generative Adversarial Networks [33], Large Graph Hashing [35], SpH [6] and SH [37] which indicate an impressive performance as the time taken to retrieved data from the database is minimal because of the short code length. As the code length increases, the performance of the competitor algorithms decreases from 8 bits to 96bits as can be shown in Table 2 and Fig. 1. It takes more time to retrieved data from a database for a query sent. The presence of unrealiable binary codes and potentially noisily-labelled samples in the data points degrade the performance RDCM [30] as the code length increases which make Geo-SPEBH has a performance difference of 0.023sec over RDCM [30] and RGC [31], as the incorporation of global and local features into [31] to construct hash function improved on the authentication of image but increase the retrieval time because in resulted to long hash code to have better authentication as can be shown in Table 2 and Fig. 1. [32] Maximised the similarity of the same class discrete hash codes and then minimised the different class discrete hash codes increases the retrieval time of information as the code length increases. It can be shown in Table 2 as Geometric similarity preserving embedding-based hashing [15] has 0.308sec performance difference over [32] when the code length is 96bits. This is because the time it takes to load data into memory is the same in Geo-SPEBH hashing [15] as each hash bucket has equal number of data points and binary hash codes, and the data points are tight with closed region using hyper-sphere. The closely competitors with our Geometric similarity preserving embedding-based hashing [15] as shown in the Table 2 are DDH [32], LGH [35] and DSH [37] which recorded 0.041sec, 0.040secs for the code length of 8bits, 0.446sec, 0.330sec and 0.320 for the code length of 64bits, and 0.568sec, 0.420sec., and 0.400sec for the code lengths of 96bits. The Geo-SPEBH [15] has a performance gain over the most closely competitor Density Sensitive Hashing [37] of 0.010sec., 0.060sec and 0.140secs for the codes lengths of 8, 64 and 96 bits respectively. Memory cost is the amount of memory or storage space occupied by the data in the data base. In Table 2, the number of bits from 8 to 96 indicate the memory cost for each algorithm occupied by data points as represented by binary hash codes. The higher the memory cost, the more time it takes to retrieved data from the database in cloud computing environment for the SIFT 1B [37] dataset for all code lengths. The low memory cost recoded by our proposed Geo-SPEBH hashing algorithm with high performance indicate that it can handle large amount of data (huge database). Table 2 gives the simulation results based on time (sec), and it can be shown that the use of tighter region, use of independent hashing function, and the balance partitioning of data points proposed by Geo-SPEBH [15] algorithm improved the speed of retrieving information from a database from a database within the shortest possible time. The proposed algorithm out-performed the compared state-of-the-art techniques with a performance difference of 0.140sec. When the code length is 96bit as compare to the most close competitor DSH [37] algorithm. Therefore, the proposed Geometric similarity preserving embedding-based hashing [15] algorithm takes less time to retrieve information from a database in cloud computing environment.

5. Conclusion

In this paper, we present a Geometric similarity preserving embedding-based hashing with a balance partitioning algorithm for improving the minimum search for each query sent for faster information retrieval from a database by distributing equal number of data points to hash buckets, and an algorithm that find and calculate the time difference with the compared algorithms. To maintained fast retrieval time performance with short code-word, we use a similarity preserving term to minimise memory cost experiment was conducted and the simulation results shows that Geometric similarity preserving embedding-based hashing outperformed the stat-of-the-art techniques as it provides faster retrieval of information. Our future research will measure the amount of bandwidth required to transfer information from source to destination.

Conflicts of Interest

I, Abubakar Usman Othman, declare no conflicts.

Author Contributions

This research was carried out by all the authors. The supervision was done by Professor Boukari Souley and Associate Professor Abdulsalam Y.G. Conceptualisation, Methodology were contributed by Abubakar U.O., Hauwa A., Maryam M., Aisha Y. U. Software: Abubakar U.O. formal analysis, investigation, validation and resources was done by all the Authors. Abubakar U.O., did the writing, draft preparation and submission of the manuscript.

References

[1] T. Danan, S. C. Surya, C. N. Rafael, and A. Leila, “A platform for monitoring and sharing of generic health data in the cloud”, *Future Generation Computer System*, Vol. 35, pp. 102-113, 2014.

[2] N. Davidovitch and R. Yavich, “The impact of mobile tablet use on students’ perception of learning processes”, *Problems of Education in the 21st Century*, Vol. 76, No. 1, pp. 29-42, 2018.

[3] S. Tatcha and M. Hitoshi, “The Internet of Things as an accelerator of advancement of broadband networks: A case of Thailand”, *Telecommunications Policy*, Vol. 42, No. 4, pp. 293-303, 2018.

[4] M. Gartner, A. Rauber, and H. Berger, “Briging structured and unstructured data via hybrid semantic search and interactive ontology-enhanced query formulation”, *Knowledge Information System*, pp. 1-32, 2013.

[5] J. Chen, C. Yuegue, E. Lia, I. L. Cuiping, and U. L. Jiaheng, “Big Data Challenges: A data Management Perspective”, *Higher Education Press and Springer Verlag Berlin Heidelberg*, Vol. 7, No. 2, pp. 157-164, 2013.

[6] Y. Wang, L. A. Kung, and T. A. Byrd, “Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations”, *Technological Forecasting and Social Change*, Vol. 126, pp. 3-13, 2015.

[7] X. Guohui and D. Linfang, “KB4Rec”, *Data Intelligence*, Vol. 23, pp. 0-2, 2019.

[8] S. Liu, Y. Wang, A. Wen, L. Wang, N. Hong, F. Shen, S. Bedrick, W. Hersh, and H. Liu, “CREATE: Cohort Retrieval Enhanced by Analysis of Text from Electronic Health Records using OMOP Common Data Model. 5”, http://arxiv.org/abs/1901.07601, 2019.

[9] D. Agrawal, P. Bernstein, E. Bertino, S. Davidson, and U. Dayal, “Challenges and Opportunities with Big Data”, *A White Paper Prepared for the Computing Community Consortium*, pp. 1-16, 2012.

[10] A. Merkys, A. Vaitkus, D. Chateigner, P. Moeck, P. Murray-rust, M. Quiros, R. T. Downs, W. Kaminsky, and A. L. Bail, “Crystallography Open Database : history, development, perspectives”, http://arxiv.org/abs/1901.07601, 2018.

[11] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi-probeLSH: Efficient indexing for high-dimensional similarity search”, In: *Proc. of International Conference on Very Large Data Bases*, pp. 950-961, 2007.

[12] W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li, “Modelling LSH for performance tuning”, In: *Proc. of the ACM Conference on Information and Knowledge Management*, pp. 669-678, 2008.

[13] Y. Gong and S. Lazebnik, “Iterative Quantisation: A procrustean approach to learning binary codes”, In: *Proc. of IEEE*

Conference on Computer Visionand Pattern Recognition, pp. 817-824, 2011.

[14] W. Liu, J. Wang, S. Kumar, and S. F. Chang, "Hashing with graphs", In: *Proc. of International Conference on Machine Learning*, 2011.

[15] U. O. Abubakar, S. Boukari, Y. G. Abdulsalam, and A. Hauwa, "Performance Evaluation of Geometric Similarity Preserving embedding-Based Hashing for Big Data in Cloud Computing", *Journal of Theoretical & Applied Information Technology*, Vol. 98, No. 3, pp. 378-390, 2020.

[16] S. Boukari, U. O. Abubakar, Y. G. Abdulsalam, and M. A. Iliya, "Performance Evaluation of Geometric Similarity Preserving", *Global Scientific Journals*, Vol. 7, No. 4, pp. 642-657, 2019.

[17] M. Raginsky and S. Lazebnik, "Locality-sensitive Binary Codes from Shift Invariant Kernels", In: *Proc. of Advance Neural Information Processing System*, pp. 1509-1517, 2009.

[18] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu, "Semi-supervised nonlinear hashing using bootstrap sequential projection learning", *IEEE Transaction on Knowledge Data Engineering*, Vol. 25, No. 6, pp. 1380-1393, 2013.

[19] X. Li, L. Gao, X. Xu, J. Shao, F. Shen, and J. Song, "Kernel based latent semantic sparse hashing for large-scale retrieval from heterogeneous data sources", *Neurocomputing*, Vol. 253, pp. 89-96, 2017.

[20] A. Chakraborty and S. Bandyopadhyay, "conLSH: Context based Locality Sensitive Hashing for Mapping of noisy SMRT Reads", *ArXiv.Org*, *q*-bio.*GN*, pp. 1-11, 2019.

[21] Y. Wang and J. Wang, "FLASH : Randomized Algorithms Accelerated over CPU-GPU for Ultra-High Dimensional Similarity Search", 2018.

[22] B. Seok, J. Park, and J. H. Park, "A lightweight hash-based blockchain architecture for industrial IoT", *Applied Sciences (Switzerland)*, Vol. 9, No. 18, 2019.

[23] M. Norouzi and D. M. Blei, "Minimal loss hashing for compact binary codes", In: *Proc. of International Conference on Machine Learning*, pp. 353-360, 2011.

[24] M. Norouzi and D. J. Fleet, "Cartesian K-means", In: *Proc. of International Conference on Computer Vision and Pattern Recognition*, pp. 3017-3024, 2013.

[25] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projection", In: *Proc. of IEEE on international conference on computer vision and pattern recognition*, pp. 484-491, 2013.

[26] L. Yueming, W. Y. Wing, Z. Ziqian, S.Y. Daneil, and P. K. Patrick, "Asymetric Cyclcial Hashing for Large-Scale-Image Retrieval", *IEEE Transaction on Multimedia*, Vol. 17, No. 8, pp. 1225-1235, 2015.

[27] R. Ye and L. Xuelong, "Compact Structure Hashing Via Sparse and Similarity Embedding", *IEEE Transactions on Cybernetics*, Vol. 46, No. 3, pp. 718-728, 2016.

[28] S. Georgia and L. George, "A survey on mathematical models, simulation approaches and testbeds used for research in cloud computing", *Simulation Modelling Practice and Theory*, pp. 1-12, 2013.

[29] S. S. Pericherla, "A Comparative Analysis of Cloud Simulators", *International Journal of Modern Education and Computer Science*, Vol. 4, pp. 64-71, 2016.

[30] L. Yadan, Y. Yang, F. Shn, H. Zi, Z. Pan, and S. H. Tao, "Robust Discrete Code Modeelling for Suppervised Hashing", In: *Proc. of International Conference on Pattern Recognition*, 2017.

[31] K. K. Ram, S. Arunav, and H. L. Rabul, "Image Authentication Based on Robust Image Hashing with Geometric correction", *Journal of Multimdia Tools and Applications*, Vol. 77, No. 19, pp. 25409-25429, 2017.

[32] C. Yan, J. Jielin, L. Zhihui, H. Zuojin, and W. Waikeun, "Supervised discrete discriminant Hashing for Image Retrieval", In: *Proc. of International Conference on Pattern Recognition*, Vol. 78, pp. 79-90, 2018.

[33] S. Jingkuan, T. He, G. Lianli, X. Xing, H. Alan, and S. T. Heng, "Binary Generative Adversarial Networks for Image Retrieval", In: *Proc. of the AAAI Conference on Artificial Intelligence*, Vol. 32, No. 1, 2018.

[34] M. Charikar and P. Siminelakis, "Hashing-based-estimators for Kernel Density in High Dimension", In: *Proc. of IEEE 58th Annual Symposium on Foundation of Computer Science (FOCS)*, pp. 1032-1043, 2017.

[35] X. Li, D. Hu, and F. Nie, "Large Graph Hashing with Spectral Rotation", In: *Proc. of the AAAI Conference on Artificial Intelligence*, Vol. 31, No. 1, 2017.

[36] J. P. Heo, L. Youngwoon, H. Junfeng, C. S. Fu, and Y. S. Eui, "Spherical Hashing: Binary Code Embedding with Hypersphere", *IEEE Transaction on Pattern Analysis and Machine Inteligence*, pp. 1-14, 2015.

[37] Z, L. C. Jin, Y. Lin, and D. Cai, "Density Sensitive Hashing", *IEEE Transactions on Cybernetics*, Vol. 44, No. 8, pp. 1362-1371, 2014.

[38] H. Junfeng, R. Regunathan, H. S. Fu, and B. Claus, "Compact Hashing with Joint Optimisation of Search Accuracy an Time", *CVPR*, pp. 753-760, 2011.

[39] H. Jegou, D. Matthijs, and S. Cordelia, "Product Quantisation for Nearest Neighbour Search", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 1, pp. 117-28, 2011.