Geometric Similarity Preserving Embedding-Based Hashing for Big Data in Cloud Computing

Boukari Souley[1], Abubakar Usman Othman[1]

[1]*Faculty of Science, Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi, Nigeria.*

***Abstract*- Indexing techniques are used on big data for efficient information retrieval from a very large and complex datasets with distributed storage in cloud computing. The availability of broad band access, mobile devices such as smartphones and tablets, body-sensor devices and cloud applications have greatly contributed to the rapid growth in data volume or big data. The existing indexing techniques are inadequate to satisfy the indexing requirements of big data. An efficient index scheme is required to meet the indexing requirement for efficient retrieval of big data. Finding approximate nearest neighbour (ANN) is essential in huge database for efficient similarity search to return the nearest neighbour of a given query. Density sensitive Hashing (DSH) achieved good performance but the discriminating information on data points are not fully utilised aside using long binary hash codes to achieve high precision-recall which slows performance as the binary code length increases and hence increase storage cost and search time. To address the aforementioned problems, this research proposes Geometric Similarity Preserving Embedding-Based hashing (Geo-SPEBH) method for improving the search accuracy and memory cost for large-scale-image retrieval. The technique aimed at preserving the underlying geometric information among data, and exploit the prior information that utilises reconstructive relationship of the data to learn compact and effective hash codes. The Geo-SPEBH makes full use of the geometric structure properties of data. An extensive experiment conducted on a cloud simulator like CloudSim should show that the proposed scheme outperforms state-of-the-art-techniques.**

***Keywords:* Big Data, Hashing, Indexing, Image, Similarity preserving, CloudSim, Interface.**

I. 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 (Thilkanathan et al., 2014). The availability of internet such as with the use of the wide spread broadband Internet access (Huang et al. 2010), coupled with these hand held devices (mobile devices), resulted to the easy collection of digital information in form of structured and unstructured (Gartner et al., 2013) 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 (Cheng et al., 2013). 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 (Doan, et al., 2009).

Big 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 data base (Agrawal et al., 2012). In the field of astronomy, taking pictures of the sky is no longer the largest part of an astronomer’s job but that pictures have been indexed in a database and other astronomers can make use of object from the indexed pictures. In business sector, purchased transactions data are efficiently stored in the cloud. 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 affective retrieval and management of big data. Data-independent (Lv, et al., 2007), (Dong, et al., 2008).Data-dependent (Gong & Lazebnik, 2011), (Liu, et al., 2011), (Abubakar, et al., 2018),(Boukari et al., 2019) hashing based approaches, and improved data-dependent hashing techniques (Norouzi & Blei, 2011), (Norouzi, et al., 2013), (Gong, et al., 2013), (Gong, et al,, 2012), were also proposed to reduced storage cost and to improve the precision rate of image retrieval in a large data base. The paramount decision when choosing and using supervised hashing techniques is based on the choice of similarity encoding approach. (Guosheng, et al., 2015), 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.

The drawback of these schemes is that the performance of the system degrade as the database increases which results to low speed, retrieval accuracy, and high search time.

Researchers are often faced with the difficulties of designing a suitable research platform when carrying out research in cloud computing (Georgia & George, 2013). Also, the cost of setting up a cloud for the benefit of conducting research by scholars on live cloud is highly exorbitant (Pericherla S. S., 2016).

II. RELATED WORKS

2.1 Indexing Techniques

Yueming et al., (2015), proposed a hashing technique that uses two hash codes of different length for stored images in the database and the queries. 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.

Ye &Xuelong, (2016), 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. Equations (1) and (2) gives the empirical fitness and the objective function respectively.

$$J(W) = \frac{1}{2}\, Tr\{W^T X_l T X_l^T W\} \qquad (1)$$

$$J_l(W) = \frac{1}{2}\, Tr\{W^T X_l T X_l^T W\} + \frac{\lambda}{2} Tr\{W^T X X^T W\} \qquad (2)$$

Where $Tr\{W^T X_l T X_l^T W\}$ is the information theoretic term. Sequential learning for hashing method is used to maximise the objective function in (2) to learn the weighted matrix W to design the hash function as in equation (3) .

$$X(X_l) = sign(W^T X_l) \qquad (3)$$

The weight matrix W and the sparse weight matrix optimally learn by minimising the objective function in equation (4).

$$(W, S) = \underset{w,s}{\arg\, min\, J_2(W,S)} \quad \dots \qquad (4)$$

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. Heo et al., (2015), 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 the spherical Hamming distance suitable to the hypersphere-based coding scheme.

The binary code embedding function $H(x)$ maps data in R^D points into the binary hash code.

Spherical Hamming distance proposed is designed to measure the distance between two binary codes computed as in equation (5).

$$d_s HD(b_i, b_j) = \frac{|b_i \oplus b_j|}{|b_i \wedge b_j|} \quad \dots \qquad (5)$$

Where $b_i \wedge b_j$ denotes the number of common +1 bit between two binary codes which can be easily computed by the AND operations. The authors employ the search time minimisation approach to balance the distribution of data points to different binary code and to also improve the search accuracy even for longer hash codes. The balance partitioning of data points is done through equation (6).

$$P_r[h_i(x) = +1] = \frac{1}{2}, x \in X, 1 \le i \le l \quad \dots \quad (6)$$

Maximally utilise the long hash codes for high precision, the hash functions were designed to be independently of one another. This independency between the hash functions is achieved through equation (7).

$$P_r[h_i(x) = +1] \,.\, P_r[h_j(x) = +1] = \frac{1}{2} \,.\, \frac{1}{2} = \frac{1}{4} \quad \dots (7)$$

The proposed scheme is generalised to a kernel one. A non-linear map $\Phi : X \rightarrow F$ from the input space X.

Jin et al., (2014), 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.

$H(x) = [h_1(x),\ h_2(x), \dots, h_L(x)]$, where $h_1(x) \in \{0, 1\}$ is the $l - th$ hash function.

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 $x_i, x_j \in R^d$ we have equation (8).

$$Pr = \left[h_l(x_i) = h_l(x_j) = 1 - \frac{1}{\pi} cos^{-1}\left(\frac{x_i^T x_j}{\|x_i\| \|x_j\|} \right) \right] \qquad (8)$$

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, where e is an error term.

In (Jin et al., 2014), despite the gains in DSH, 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 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. Where in the proposed Geo-SPEBH, hypersphere-based hashing function are used to encode proximity regions in high-dimensional spaces. The proposed method takes into account, the distribution of data points for efficient hash functions design.

$$SSE = \sum_{p=1}^{k} \sum_{x \in S_p} \left\| x - \mu_p \right\|^2 \quad (9)$$

Where μ_p is the representative point of the $p - th$ group S_p.

The use of hypersphere improves the performance of search accuracy (precision) as the code length increases. 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. Here the number groups decide the maximum length of code generate as in equation (10).

$$k = \alpha L \quad (10)$$

Where L is the code length and $\alpha$is a parameter.

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. Thenearest neighbours matrix 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. The median plane is given as equation (11), whereas the hash function with respect to the plane is as equation (12).

$$W_{ij} = \begin{cases} 1, if\ \mu_1 \in N_r(\mu_j) or\ \mu_j \in N_r(\mu_i) \\ 0, otherwise \end{cases} \quad (11)$$

Where $N_r(\mu_i)$ denotes the set of r nearest neighbours of μ_i.

$$(x - \frac{\mu_i + \mu_j}{2})^T (\mu_i - \mu_i) = 0 \quad (12)$$

$$h(x) = \begin{cases} 1\ if\ W^T \geq t \\ 0\ otherwise \end{cases}$$

Where $w = \mu_1 - \mu_2$, $t = \frac{\mu_i + \mu_j}{2})^T (\mu_1 - \mu_2)$

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 is computed by equation (13).

$$P_{i0} = \frac{|T_{i0}|}{|T_{i0}| + |T_{i1}|}, P_{i1} = \frac{|T_{i1}|}{|T_{i0}| + |T_{i1}|} \quad (13)$$

Where T_{i0} and T_{i1} are data points partitions, and P_{i0} and P_{i1} are projections.

When there is huge database, computing time increases, as such, the entropy is estimated by using the centre of the group and assign a weight based on the group's size. This is achieved a significant reduction in time for computing entropy, in equation (14) is used.

$$P_{i0} = \sum_{s \in c_{i0}} V_{st}, P_{i0} = \sum_{t \in c_{i1}} V_t \quad (14)$$

$$SSE = \sum_{p=1}^{k} \sum_{x \in S_p} \left\| x - \mu_p \right\|^2 \quad (15)$$

The k-means with p iterations to generate αL groups requires o($\alpha Lpnd$).

To find the $r -$adjacent groups, $((\alpha L)^2 (d + r))$is required.

$0(\alpha Lrd)$ for each pair of adjacent groups, generates the projections and the intercept.

To compute the entropy for all the candidate projections, $o((\alpha L)^2 dr)$ is required.

The top L projections can be found within o$(\alpha Lrlog(\alpha Lr))$ and the binary codes for the data points can be obtained in $o(Lnd)$. This shows that the computational complexity of DSH at the training phase is dominated by the k-means clustering algorithm. An efficient hashing scheme should handle large amount of data size but DSH fails to improve as the code length increases. Therefore, an efficient hash function is desirable.

The computational complexity of DSH at the testing phase with a given query point needs $o(Ld)$ to transform the query point into a binary code.

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 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.

The body literature is substantial on data-independent and data-dependent hash-based indexing methods that uses projection information and learning data for big data indexing in cloud computing, but it's limited on those that utilises discriminative information in images to preserve similar data points. Despite their successes, the above mentioned techniques are limited in that the geometric information of data are not effectively and adequately preserved while others have neglected the potential use of the structural properties of data for learning compact and effective hash codes for efficient hash-based big data retrieval in mobile cloud computing environment. The long hash codes consumes large memory of the stored data in the database thereby increasing the storage cost while the short hash codes gives unsatisfactory performance in terms of precision.

The Geo-SPEBH aim at overcoming the drawback of data-independent based hashing methods and data-dependent based hash methods. To guarantee the performance Geo-SPEBH will increase as the code length increases, Geo-SPEBH adopt the framework as DSH. While DSH 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 confirm to be very discriminative, and ensure that fewer features are inserted into the hash table. By using fewer features into the hash table, the computational cost and memory cost will be greatly reduced. The geometric properties of the principal components of features are found to be robust to handle translation and rotation effect (Umarani Jayaraman, 2013).

III. RESEARCH METHODOLOGY

3.1 The proposed System

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 Geo-SPEBH 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 databasewhich 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 equation (1)

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

Where k is the length of the binary hash code.

Figure 1 gives the conceptual framework of the proposed system. The working principle of the proposed 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.

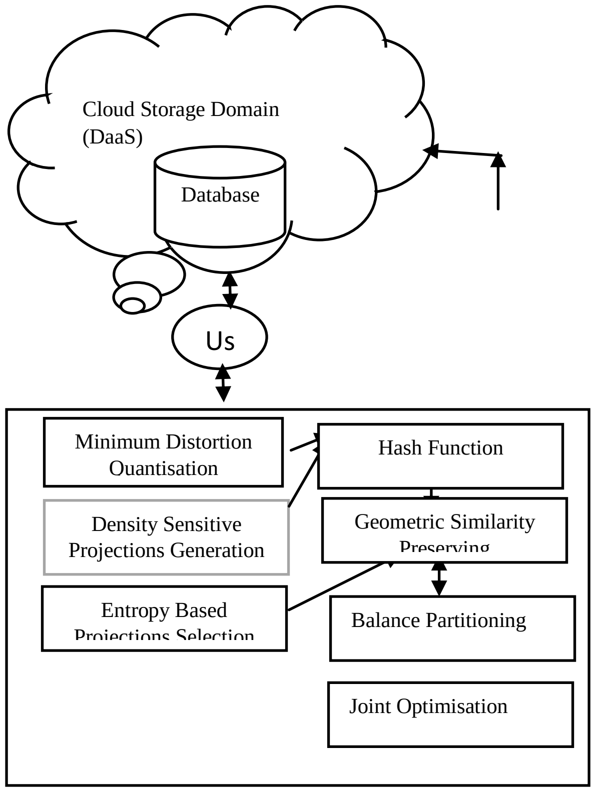
Cloud Storage Domain (DaaS)
Database
Us
Minimum Distortion Ouantisation
Hash Function
Density Sensitive Projections Generation
Geometric Similarity Preserving
Entropy Based Projections Selection
Balance Partitioning
Joint Optimisation

Figure 1. Conceptual framework of the proposed System

1. Firstly, features are extracted
2. Secondly, a hashing function to project each sample data point into compact binary hash codes was used. The hash function depends on the distribution of data to generate hash table.
3. Thirdly, the geometric similarity preserving component was use to preserve the similarity among the data points so that the original structure property of the data points are retained.
4. Fourthly, the balance partitioning component was used to distribute data points equally to each binary code. This is achieved if the hash functions are independent.
5. Fifthly, then optimise the search accuracy and search time simultaneously by incorporating the similarity preserving and the balanced partitioning components.
6. Then the data are stored in the data storage as a service (DaaS) provided by service providers in cloud computing environment.

3.1.1 Geometric Similarity Preserving

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 of data. Extracting the similarity between the two data samples as Q_{ij} from the 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 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 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). Hamming distance is minimised the by taking the absolute values of the of the similarity term as in equation (7) (Junfeng et al., 2011).

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

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

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

whereQ_{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. The Hamming distance to minimised with respect to:$yi \in \{0,1\}^k$

$$\sum_i yi = 0 \tag{8}$$

$$\frac{1}{n}\sum_i yiyi^T = I \tag{9}$$

Where the constraints (8) require 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$||Y_i - Y_j||^2$. $Y_i$andY_j are the similar hash codes.

The algorithm for similarity preserving is given in figure 2.

Algorithm 1: Similarity Preserving

```
1.  Start
2.  Input: n training sample dataset $X = \{x_1, x_2, x_3, \dots, x_n\}$ in the
    database; $w_{ij}$ the similarity between $x_i$ and $x_j$ //$u_p$ and $u_q$ are the
    hash codes for $x_i$ and $x_j$//; c, Sim
3.  Sum = 0, Ham = 0
4.  $for\ i = 1\ to\ c$//c the number of iteration for $i$//
5.  $for\ j = 1\ to\ V$//$V$ is the number of iteration for $j$//
6.              Get $W(i,j)$
7.              Sum = Sum + $i$
8.  $j = j + 1$
9.  $if\ i \le c$ goto step 6
10.                 end if
11.           end for
12. Sim = Sum
13.                break;
14.  //minimising Hamming distance//
15. $for\ i = 1\ to\ c$
16. $for\ j = 1\ to\ c$
17.             Get $y(i), y(j)$; //$u_p$ and $u_q$ are similar binary hash
    codes//
18.   Ham = Ham + $(y(i) - y(j))$**2
19. $j = j + 1$
20. $if\ j \le c$ goto step 17
21.                   end if
22. $i = i + 1$
23. $if\ i \le c$ goto step 17
24.                        end if
25.                    end for
26.             end for
27. SimM = Ham
28.  Print Sum, SimM//output//
29.  Stop
```

Figure 2. Algorithm for Similarity Preserving

3.1.2 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 equation (4) 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 equation (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 equation (4) and (5) is used to balance the partitioning of data points for each bit.

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

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

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

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

$$Pr[Bi \cap Bj] = Pr[Bi] \,.\, \mathrm{Pr}[Bj] \quad (10)$$

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

The intersection is the equal chance of the code bit being 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 a

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

Balance partitioning for distributing data points to different binary codes by independent hash functions is in figure 3.

Algorithm 2: Balanced 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
```

Figure 3. Algorithm for Balance Partitioning

3.1.3 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 proposed system is formulated and is responsible for the simultaneous optimisation of the search accuracy and search time. A parametrisation of a linear function is performed for easy optimisation, and a relaxation is perform.

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, $N = \frac{N}{2^M}$ (Junfeng et al., 2011) equation (11)

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:

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

The similarity preserving term and the balance partitioning are incorporated together for simultaneous improvement in search accuracy and search time, (Junfeng et al., 2011).

3.1.3. 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), Junfeng et al., 2011.

The joint optimisation algorithm is presented in figure 4.

Algorithm 3: Joint Optimisation

```
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//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. $if\ j \leq b$ goto step 55
37. i = i + 1
38. $if\ i \leq a$ 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. $if\ j \leq b$ goto step 45
51. $i = i + 1$
52. $if\ i \leq a$ 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. $if\ i \leq a$ goto step 78
61.        end if
62.  end for
63.  Stop
```

Figure 4. Joint Optimised Algorithm for Search Accuracy and Time

IV. EXPERIMENTAL SETUP

This section provides and avenue for evaluating the proposed technique and make a comparison to the state-of-the-art-techniques.

4.1 Performance metrics

The Geo-SPBH will be 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 **SIFT 1B** dataset. The MAP measures the average of precision scores for the queries. It is the area under the precision-recall curve for a set of queries. A large value of MAP will indicate a better performance.

4.2 Evaluation Techniques

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

1. *Density Sensitive Hashing:* 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 (Jin et al., 2014). thismethod is an extension of LSH.
2. *Locality Sensitive Hashing:* 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 used for projection generation are randomly sample from a p-stable distribution. It is an unsupervised technique that apply to change detection domain (Datar, et al., 2004)
3. *Kernerlised Locality Sensitive Hashing:* KLSH generalises the LSH to the kernel space (Kulis & Grauman, 2009).
4. *Shift-Invariant Kernel Hashing* **(**Raginsky & Lazebnik, 2009)**:** The SIKH is based on the random feature hashing for approximating the shift-invariant kernels.
5. *Geo-SPEBH:* This is the proposed method to be compared with the above mentioned algorithms.

4.3 Programming tools used

All the experiments were conducted and run on a 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 a output size of 300 each and length of 5000.

4.4 Results

To generate 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.

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

4.4.1 Implemented Results

SIFT 1B dataset from simulation results carried out on the existing algorithm implemented on the proposed system and the results were compared. The proposed Geo-SPEBH algorithm is implemented and compared the result with state-of-the-art-techniques. The based algorithm (DSH) and other existing state-of-the-art techniques are run with the new components and the results are obtained and compared. The results obtained shows that the proposed system outperform the existing techniques based on the mean average precision results as shown in table 2.

Table 2. Simulation Results for the Proposed and Existing Methods

		SIFT 1B Mean Average Precision (%)					
		Code length (bits)					
METHODS	8	16	32	48	64	80	96
LSH	0.0459	0.0687	0.0895	0.1005	0.1515	0.1534	0.1719
DSH	0.0600	0.1250	0.3434	0.1300	0.1507	0.1616	0.2501
SIKH	0.0503	0.0509	0.0641	0.0687	0.0723	0.0898	0.1235
KLSH	0.0547	0.0590	0.0900	0.1109	0.1194	0.1387	0.1466
GSPEBH	0.0688	0.1299	0.3906	0.1414	0.1677	0.1916	0.2605

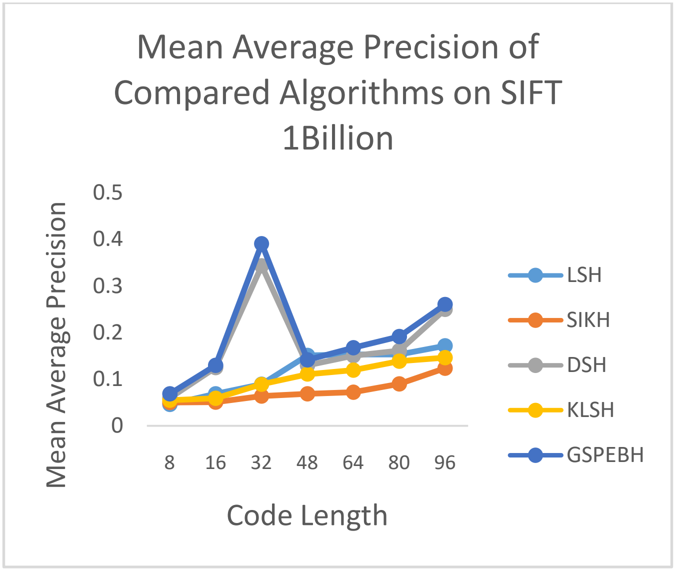

Figure 5. The Mean Average Precision of all the Algorithms on SIFT 1B Dataset.

4.2 Discussion

The **SIFT BM** datasetis 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 the old algorithm (DSH) with varied number of bits, 8, 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.

V. CONCLUSION

It can be seen that the data-independent 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. From table 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 SIFT 1B 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 2 gives the Mean Average Precision results for the SIFT dataset for all the compared methods. Given 0.3 Mean Average Precision 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.

Further research should be directed towards finding a solution to balancing the trade-off between precision-recall, and the measure the performance based on search time. Furthermore, the data collected from different sources in a raw form such as student records, health records, mathematical and statistical analysis cannot be effectively analysed. An advanced technique is required so that data can be extracted from different sources to structured them in a format that can be used for analysis.

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]. Huang, Z., Heng, T. S., & Shao, J. (2010). Bounded Coordinate System Indexing for Real-time. *ACM Transactions on Information Systems*, *10*(10), 1-32.

[3]. Gartner M, Rauber, A., & Berger, H. (2013). Briging structured and unstructured data via hybrid semantic search and interactive ontology-enhanced query formulation. *Knowledge information system*, 1-32.

[4]. Chen, J., Yuegue, C., Lia, E., Cuiping, I. L., & Jiaheng, U. L. (2013). Big Data Challenges: A data Management Perspective. *Higher education press and springer verlag Berlin Heidelberg*, *7*(2), 157-164.

[5]. Doan, A, N. J., Baid, A., Chai, X., Chen, F., Chen, T., . . . Vuong, B. Q. (2009). The case for a structured approach to managing unstructured data. *In Proceeding of the 4th Biennial Conference on Innovative Data Systems Rsearch*.

[6]. Agrawal, D., Bernstein, P., Bertino, E., Davidson, S., & Dayal, U. (2012). Challenges and Opportunities with Big Data. *A white paper prepared for the Computing Community Consortium*, 1-16.

[7]. Lv, Q., Josephson, W., Wang, Z., Charikar, M., & Li, K. (2007). Multi-probeLSH: Efficient indexing for high-dimensional similarity search. *Proceedings of international conference on Very Large Data Bases*, 950-961.

[8]. Dong, W., Wang, Z., Josephson, W., Charikar, M., & Li, K. (2008). Modelling LSH for performance tuning. *In Proceedings of the ACM conference on information and knowledge management*, 669-678.

[9]. Gong, Y., & Lazebnik, S. (2011, June). Iterative Quantisation: A procrustean approach to learning binary codes. *Proceedings of IEEE conference on computer visionand pattern recognition*, 817-824.

[10]. Liu, W., Wang, J., Kumar, S., & Chang, S.-F. (2011). Hashing with graphs. *In International Conference on Machine Learning*.

[11]. Abubakar, U. O., Boukari, S., Abdulsalam, Y. G., Iliya, M. A., (2018). Geo-Cyclical Structure-Based Hashing, World Journal of Engineering Research and Technology, 203-217

[12]. Boukari, S., Othman, U. A., Abdulsalam, Y. G., Iliya, M. A., (2019). Performance Evaluation of GeometricSimilarity PreGlobal Scientific Journals, 7(4), 642-657.

[13]. Norouzi, M., & Blei, D. M. (2011). Minimal loss hashing for compact binary codes. *Proceedings of international conference on machine learning.*, 353-360.

[14]. Norouzi, M., & Fleet, D. J. (2013, June). Cartesian K-means. *Proccedings of international conference on computer vision and pattern recognition*, 3017-3024.

[15]. Gong, Y., Kumar, S., Rowley, H. A., & Lazebnik, S. (2013, June). Learning binary codes for high-dimensional data using bilinear projection. *Proceedings of IEEE on international conference on computer vision and pattern recognition*, 484-491.

[16]. Gong, Y., Kumar, S., Vernma, V., & Laxebnik, S. (2012). Angular Quantisation Based Binary Codes for Fast Similarity Search. *Proceedings of NIPS* , 1205-1213.

[17]. Guosheng, L., Chunhua, S., & Anton, V. d. (2015). Supervised hashing using graph cuts and boosted decision trees. *IEEE transactions on pattern analysis and machine intelligence, 37*(11).

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

[19]. Pericherla, S. S. (2016, April). A Comparative Analysis of Cloud Simulators. *International Journal of Modern Education and Computer Science, 4*, 64-71.

[20]. 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.

[21]. Ye, R., & Xuelong, L. (2016, March). Compact Structure Hashing Via Sparse and Similarity Embedding. *IEEE Transactions on Cybernetics, 46*(3), 718-728[4] Aguilera M, K., Golab, W., & Shah, M. A. (2008). A practical scalable distributed b-tree. *Proceedings of the VLDB Endowment, 1*(1), 598–609.

[22]. Heo, J.-P., Youngwoon, L., Junfeng, H., Shih-Fu, C., & Sung-Eui, Y. (2015). Spherical Hashing: Binary Code Embedding with Hypersphere. *IEEE Transaction on Pattern Analysis and Machine Inteligence*, 1-14.

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

[24]. Umarani Jayaraman, S. P. (2013). Use of geometric features of principal components for indexing a biometric database. *Mathematical and computer modelling, 58*, 147-164.

[25]. Junfeng, H., Regunathan, R., Shih-Fu, h., & Claus, B. (2011). Compact Hashing with Joint Optimisation of Search Accuracy an Time. *CVPR*.

[26]. 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.

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

[28]. Raginsky, M., & Lazebnik, S. (2009). Locality-sensitive Binary Codes from Shift Invariant Kernels. *Proceedings of Advance Neural Information Processing System*, 1509-1517.