Journal of Theoretical and Applied Information Technology
15th February 2020. Vol.98. No 03
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PERFORMANCE EVALUATION OF GEOMETRIC SIMILARITY PRESERVING EMBEDDING-BASED HASHING FOR BIG DATA IN CLOUD COMPUTING.

ABUBAKAR USMAN OTHMAN, BOUKARI SOULEY, ABDULSALAM YA'U GITAL, HAUWA ABUBAKAR.

Faculty of Science, Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi, Nigeria.

E-mail: othman80s@yahoo.com

ABSTRACT

Approximate nearest neighbour (ANN) search has been favourable for large-scale information retrieval in the recent past, and many hashing techniques for ANN have been proposed for retrieval of data in a large database, given a query. Hashing based indexing techniques are being mostly favoured for similarity search from huge database because of its efficiency in retrieval accuracy and low memory requirements. The long code length of randomised hashing based indexing techniques gives good precision but required more computational cost and high memory cost. DSH uses K-means algorithm to partition n data points into k groups for quantisation of data. This paper addresses the problem of long hash codes, computational cost, long convergent time and high memory requirements to achieve efficient similarity searching. Experiment was setup and Geo-SPEBH was evaluated on SIFT 1B based on MAP, precision-recall metrics and Geo-SPEBH outperformed the state-of-the-art techniques.

Keywords- *Hashing, Similarity Preserving, Binary Codes, Indexing, Bid Data, Cloud Computing.*

1 INTRODUCTION

The rapid advances of communication devices such as smartphones used in a variety of applications [1], coupled with the affordability and availability of broadband internet [2], provides an easy collection of digital information in form of structured and unstructured [3] data, has contributed to accruing of huge amount of data or big data. These large-scale databases give a big problem in terms of scalability, and needed to be indexed for efficient retrieval and management of data. Information extracting technologies, in the recent past, have been employed to extract meaningful information from raw or unstructured data [4]. The unstructured data being generated data need to be structuralised by extracting information so that the extracted information can be used by analysis algorithm [5]. Technologies are needed for proper processing of these data to open new discoveries and knowledge [6]. Less time and cost are needed indexing moving objects to analyse big data with indexing techniques thus efficiently indexing of big data results to reduce time while still tolerating high cost when designing such indexing methods [7]. Effective schemes for indexing, updating and querying this dataset were developed. Such effective schemes are evident in the field of car tracking [8], gaming engines [9], and tracking of mobile phones [10]. Content based image indexing and retrieval, video indexing audio indexing aims at obtaining a structured indexing of the original video content and get familiar with its embedded semantics just as with human beings [11].

Binary code embedding techniques provide efficient alternative for similarity search and compact data representations that is most suited to handle huge databases in the field of information retrieval, computer vision and pattern recognition.

The similarity search and nearest neighbour search such as the tree-based indexing techniques have been numerously proposed by [12], [13], [14], [15], [16], [17], [18]. The tree based indexing techniques is efficient in low dimensional data and are not scalable to high-dimensional datasets. For these, the vision community has in the recent past put a lot of efforts to the challenges of learning similarity preserving binary codes for representing large-scale database as a solution to the tree based methods. ANN methods [19], [20], [21], are proposed for approximate nearest searches. Many other hashing approaches [22], [23], [24], [25], [26],

[27], [28], [29],[30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], and [45], have also been proposed to provide solutions to high dimensional datasets. High-dimensional data points are encoded into binary codes based on binary code embedding methods to achieve a higher scalability through compact representation of data and efficient indexing technique. In hashing based indexing techniques, similar data points are mapped to similar binary codes to achieve low memory requirement.

Hashing based indexing techniques can broadly be classified into data-independent indexing techniques [26], [35], [46], [27], also known as randomised hashing techniques because they do not make any prior assumption about the data distribution. These categories of hashing based indexing methods generate their projections randomly and, the data-dependent binary code embedding techniques [47], uses the data distribution to generate the hash function.

However, the long hash codes generated by data-independent based indexing techniques is usually very high and thus very expensive foe large scale database. Long codewards results to high storage cost. This reduce the number of data stored in a database. If the data to be stored is very large and the memory cannot longer accommodate the binary hash codes generated by the randomised indexing approaches, the alternative is to frequently access a distributed system which is of course slower than the direct memory access [48]. When this occurs, query may collapse due to long response time. On the other hand, the data-dependent, also known as learning-based hashing techniques makes use of the structure of the data with respect to data distribution to generate the hash functions (hash table).

To address the problem of long hash codes and maintaine a trade-off between precision-recall, for efficient retrieval accuracy and storage requirements, we propose a similarity preserving and independent hashing function scheme. The data points are evenly distributed in a balanced proportion to binary hash codes while preserving the similarity between the data points.

The remaining of this paper is organized as follows. Section 2 presents literature related to the work, section 3 presents the methodology used while section 4 presents the results of the proposed system. We presents the discussion and conclusion in section 5.

2 RELATED WORKS.

The most popular scheme among the class of randomised hashing based indexing scheme is the LSH [20], which has since been developed to many variants such as the $l_p$norm LSH [30], KLSH [47], learning to hash with binary [49], locality sensitive binary code [35], uses random linear projections to map to map data into binary codes. In data-independent hashing techniques, the Hamming distance between two binary codes asymptotically approaches the distance in the original feature space as the code length increases [50], which results to generating of long codes to achieve satisfactory performance.

The data-independent hashing techniques on the other hand learn compact binary codes from training data. It generates its projections using the prior knowledge of data distribution. The data-dependent hashing techniques can be grouped into three category according to the level of semantic labels used: supervised, semi-supervised and unsupervised hashing methods. The supervised data-dependent hashing methods learn their hash functions using labelled data while the supervised hashing techniques generated their projections using unlabelled information of training data that seek to propagate neighbourhood similarity of samples from a certain metric space into the hamming space [32], [51], [52], [53], [54]. The unsupervised category of learning based indexing techniques does not require label information of training data in projection generating process but tries to keep the similarity information between training samples in the original space as the data points samples are projected into the hamming distance space. Categories of these unsupervised hashing algorithms also includes KMH [56], which normally give satisfactory performance. The GSPEBH method exploits the structure property of the data to efficient binary codes. SH [32], uses the graph partitioning to generate good binary codes. [50], proposed a deep convolutional neural network to be able to encode similar images to similar binary codes by minimising the distance between the Euclidean distance to Hamming space. [57], proposed a Bit-Scalable Deep Hashing with Regularised Similarity Learning for Image Retrieval and Person Re-identification to address the problem of neglecting the significance level of different bits and restricting their practical flexibility by analysing the training images into a batch of triplet samples. The triplet samples are then maximised between matched pairs and mismatched pairs in the Hamming space.

[58], proposed an asymmetric hashing technique based on random projection to improve the search accuracy of images. Their proposed method, uses the real-valued output of the hash function to compute a weighted Hamming distance based lower bound of the Euclidean distance between the query and the stored data in the database. [59], made an improvement by generalising to a broader class of binary embedding by computing the weighted Hamming distance based on the lower bound of Euclidean distances between the real-valued outputs of hash function of the query and the binary hash codes of the stored data in the database to improve on the precision of retrieved information in the database. [44], proposed an Asymmetric Cyclical hashing based indexing technique for efficient retrieval of large-scale images in a database by using two distinct hash codes of different length for query and stored data in a database that is the asymmetric cyclical hashing. They seek a compact hash code to reduce the storage requirement while the long hash code is used for the query. Information retrieval is performed by computing the Hamming distance of the long hash code and of the query and the cyclically concatenating compact hash codes of the stored data for efficient retrieval in terms of precision and recall. The challenges of these methods are that they incur high computational cost in addition to the computation of the Hamming distance between the stored data in the data base and the query. In [60], a density sensitive hashing technique was proposed to generate projections by exploring the structure of data to generate hash function and employ only those projective functions that best agreed with the data distribution.

3 THE PROPOSED SYSTEM.

Here we present our proposed system and its working principle. The proposed system is composed of four components that perform a particular function to achieve retrieval accuracy and minimal storage requirement. The purpose of learning hashing-based methods is to use the mapping function 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 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. 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 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 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 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.

3.1 Hash Function

This component of the proposed system, is responsible for hashing 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 proposed system compresses the original descriptor of the stored images in the database to a low-dimensional k-bit compact binary hash codes with high compression ratio for small storage cost. 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 dependent on the data points. We use K hash functions based on hypersphere to preserve the similarity among data point samples. The constructed hash function is then used 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 utilised 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. 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 neighbour from a query point ANN search, closed regions 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 has 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) [28], is where the hash codes are generated.

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

We use 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 $\{h_x\}_{k=1}^{K}$

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

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 mapped into same bucket with high probability of having a 50% chance of becoming one (1) by defining independence of each bit. Equation (4), is used to balance the partitioning of data points for each bit for each data point x [28].

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

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

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

3.2 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 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 by taking the absolute values of the of the similarity term as in equation (7) [29].

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

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

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

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. The Hamming distance to minimised with respect to:

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

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

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 and Y_j are the similar hash codes.

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 given the opportunity of becoming 0 or 1 since binary digits are represented by zeros (0's) and ones (1's). 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.

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.

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

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

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.

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

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

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

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 as

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

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 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 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, $N = \frac{N}{2^M}$ [29], equation (11) Here, the search accuracy is improved by minimising the Hamming distance between similar data points.

$Q(y) = sum\ of\ samples\ of\ x\ from\ i\ to\ N \quad + sum\ of\ samples\ of\ x\ from\ j\ to\ N.$
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 balance partitioning are incorporated together for simultaneous improvement in search accuracy and search time, [29].

Algorithm 1: Geo-SPEBH

```
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; TP; FP; FN; Get equation (7);
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
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.             end if
52.         $i = i + 1$
53.             $if\ i \le a$ goto step 65
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. //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.                 $if\ j \le k$ goto step 89
73.                 end if
74.             end for
75.         end for
76. Print JointO
77. //computing precision//
78. //TP is the true positive i.e. document relevant to the user//
79. //FP is the false positive i.e. documents retrieved but not needed//
80. Sum = 0
81. Get TP, FP, FN
82. Pr = sum + (TP/(TP + FP))
83. Re = sum + (TP/(TP + FN))
84. Print Pr, Re//output Precision, Recall//
85. //computing mean average precision//
86.         Sum = 0
87.           Get R, L//L is the number of the retrieved data//
88.             for $i = 1\ to\ R$
89.              Get y(i)
90. MAP = sum + $y(i)(\ P(r) * \delta(r))/L)$
91.                 $i = i + 1$
```

```
92.                 if $i \leq R$ goto step 87
93.                  end if
94.                end for
95. Print MAP//output Mean Average
    Precision//
96.  Stop
```

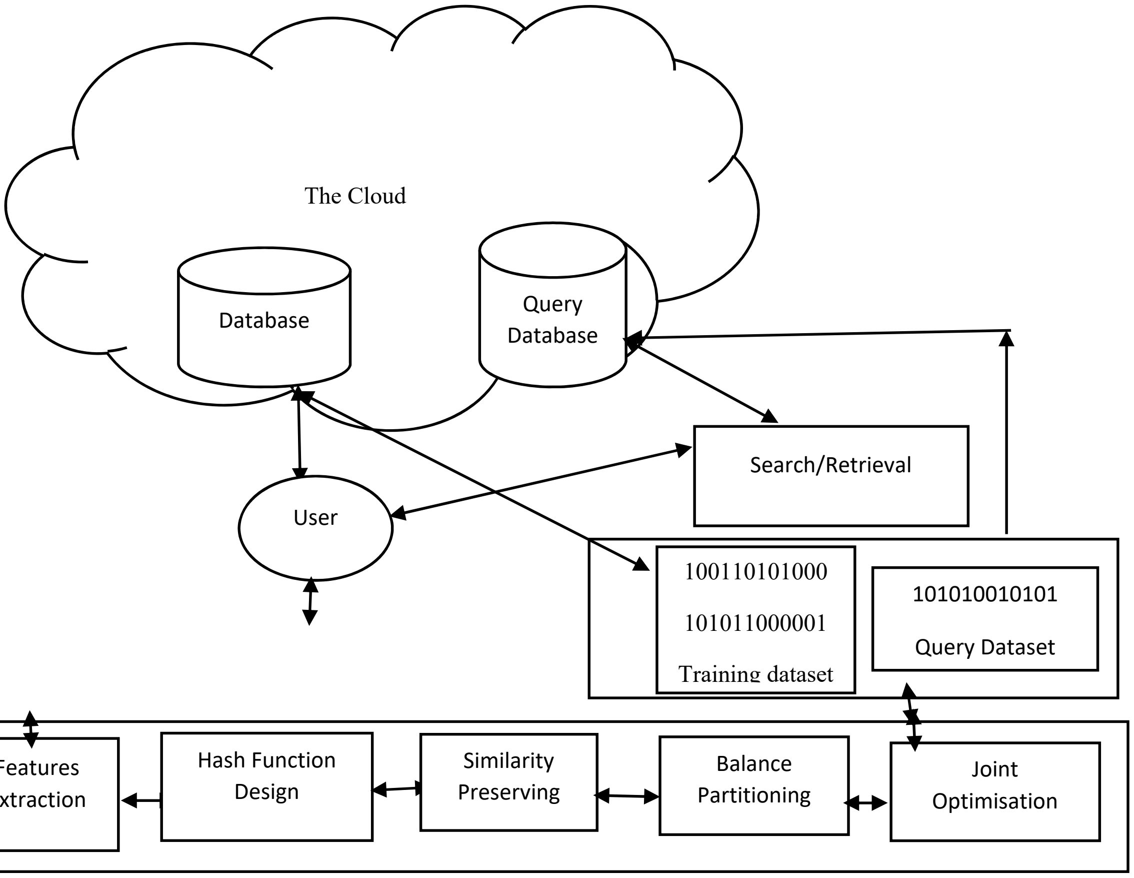

Figure 1. Architecture Of The Proposed System Model

4 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) and precision/recall 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, and the precision/recall measures the precision by recall per code lengths for each code bits of 8, 16, 32, 48, 64, 96.

4.2 Evaluation Techniques.

The algorithms used in the evaluation of the proposed system are the DSH, SHD, AGH, SpH, 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 [59]. It is an extension of LSH.

Spherical Hamming Distance: SHD is a hashing based technique that uses spherical Hamming distance [28].

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 [60].

Spectral Hashing: SpH is a classic approach that quantised the values of analytical Eigen functions computed along the principal component analysis of the data [32]. 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 [52].

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 Preliminary Results

To generate discriminative binary hash codes that need 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.

4.5 Tools to be used

The dataset used for the evaluation of our technique is the **SIFT-1B-128D [61],** which contains one billion SIFT features. Each of the SIFT features is represented by a 128-dim vectors.

Table 1. Sift 1billion Data Set

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

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.

4.6 Simulation Results

We use the SIFT 1B dataset to implement our USPH algorithm and compared our result with state-of-the-art-techniques. The existing algorithms are run and results are obtained and recorded. The base algorithm (DSH) is run without the new components and results are obtained. The new algorithm is run with the new components and the results are obtained and compared with the base algorithm and other existing state-of-the-art techniques. The results obtained show 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
DSH	0.0600	0.1250	0.3434	0.1300	0.1507	0.1616	0.2501
SHD	0.0245	0.0439	0.0685	0.0945	0.1000	0.1547	O.1843
SpH	0.1824	0.1131	0.2588	0.1448	0.1589	0.1696	0.1711
AGH	0.0521	0.0895	0.1445	0.1546	0.1585	0.1598	0.1653
PCAH	0.0555	0.0695	0.1958	0.1250	0.1605	0.1698	0.1742
Geo-SPEBH	0.0688	0.1299	0.3906	0.1414	0.1677	0.1916	0.2605

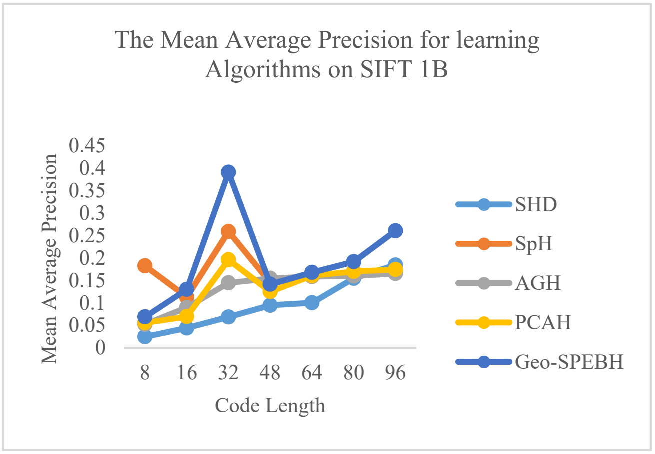

Figure 2. The Mean Average Precision Of Data-Dependent Hashing Algorithms On SIFT 1B Dataset.

Table 3. Simulation Results For The Proposed And Existing Methods

		SIFT 1B Precision					
		Code length (bits)					
METHODS	8	16	32	48	64	80	96
DSH	0.0600	0.1250	0.3434	0.1300	0.1507	0.1616	0.2501
SHD	0.0245	0.0439	0.0685	0.0945	0.1000	0.1547	O.1843
SpH	0.1824	0.1131	0.2588	0.1448	0.1589	0.1696	0.1711
AGH	0.0521	0.0895	0.1445	0.1546	0.1585	0.1598	0.1653
PCAH	0.0555	0.0695	0.1958	0.1250	0.1605	0.1698	0.1742
Geo-SPEBH	0.0688	0.1299	0.3906	0.1414	0.1677	0.1916	0.2605

Table 3. Simulation Results For The Proposed And Existing Methods

		SIFT 1B Recall (%)					
		Code length (bits)					
METHODS	8	16	32	48	64	80	96
DSH	0.4060	0.3838	0.3550	0.3500	0.3454	0.3400	0.3323
SHD	0.2505	0.3788	0.3854	0.3900	0.4545	0.4860	O.4880
SpH	0.4545	0.4055	0.4045	0.4011	0.4145	0.4400	0.4520
AGH	0.3000	0.3113	0.3380	0.3455	0.3600	0.3775	0.3855
PCAH	0.3900	0.3865	0.3845	0.3825	0.3810	0.3785	0.3735
Geo-SPEBH	0.5383	0.5200	0.5064	0.5179	0.5153	0.5152	0.5150

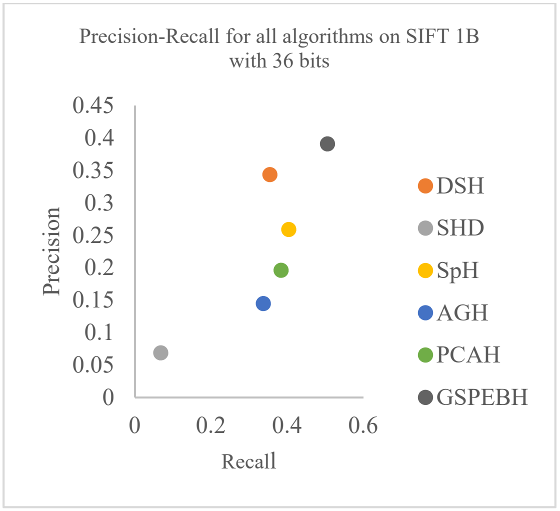

Figure 3. The Precision/Recall Of Algorithms On SIFT 1B Dataset With Code 32 Bits.

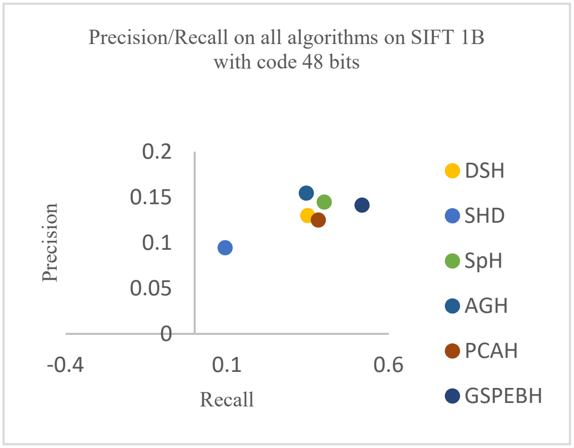

Figure 4. The Precision/Recall Of Algorithms On SIFT 1B Dataset With Code 48 Bits.

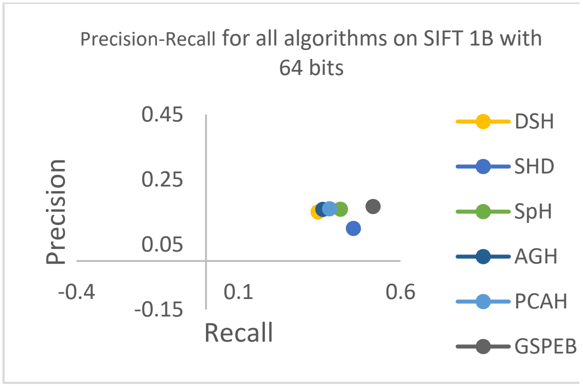

Figure 5. The Precision/Recall Of Algorithms On SIFT 1B Dataset With Code 64 Bits.

4.8 Discussion

The **SIFT 1B** 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 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.

In Table 2, 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. The data-dependent techniques, PCAH, SpH, SHD, and AGH semi-supervised technique which is DSH recorded a high MAP performance when the code length is short but drops when the code length increases. From table 3 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 MAP results for the SIFT 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.

The higher the value of precision-recall, the better the performance in terms of efficiency in retrieval accuracy. From figure 3, it can be seen that at the code length of 32 bits, the Geo-SPEBH which is the proposed system recorded a precision/recall rate of 0.3906 and 0.5064 respectively as against 0.3434 and 0.3550 recorded by DSH method which is our based research paper. It clearly shows that the proposed system has a better precision/recall which gives better retrieval accuracy than the compared algorithms. Figures 3, 4 and 5, presented the simulation results for all the algorithms with respect to codes bits of 32, 48 and 64. It is well suited to point out that the utilisation of the geometric properties of data in generating hash functions, preserving similarity between the data points and the balance partitioning of data to have equally distributed data points in each hash table by the proposed system produce better performance to the compared algorithms for all the code lengths.

The codes lengths represented by code lengths of 32, 48 and 64 bits represented the memory (storage) requirement for each data stored in the database in bits.

5 CONCLUSION

The performance has showed 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 PCAH, DSH, SHD, SpH 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 structure them in a format that can be used for analysis.

REFERENCE

[1] T. S. C. Danan, N. Surya, C. Rafael, and A. Leila,."A platform for monitoring and sharing of generic health data in the cloud", *Future generation computer system, 35*, 2014, 102-113.

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

[3] 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*, 2013, 1-32.

[4] A. H. Doan, J. F. Naughton, A. Baid, X. Chai, F. Chen, T. Chen, E. Chu, P. DelRose, B. Gao, C. Gokhale, J. Huang, W. Shen, and B. Q. Vuong, "Information extraction challenges in managing unstructured data", *ACM SIGMOD*, 37(4), December, 2008, 14-20.

[5] C. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalam, "A survey of web information extraction system", *IEEE Transaction on Knowledge and Data Engineering, 18*(10), 2006, 1411-1428.

[6] J. Cheng, 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, 7*(2), 2013, 157-164.

[7] S. Kadiyala, and N. Shiri, "A compact multi-resolution indedx for variable length queries in time series database", *Knowledge information system, 15*(2), 2008, 131-147.

[8] C. Jensen, and S. Pakalnis, "TRAX—real-world tracking of moving objects", *n: VLDB*, 2007, 1362-1365.

[9] W. White, A. Demers, C. Koch, J. Gehrke, and R. Rajagopalan, "Scaling games to epic proportion. *In SIGMOD, June, 2007, 31-42..*

[10] E. A. Anderson, "Shakra: tracking and sharing daily activity levels with unaugmented mobile phones", *mobile network application, 12*(2-3), 2007, 185-199.

[11] B. B. Meshram, and G. P. Gaikwad, "Different indexing techniques", *International Journal of Engineering Research and Application, 3*(2), April, 2013, 1230-1235.

[12] M. Muja, and D. G. Lowe, "Fast Approximate Nearest Neighbours with Automatic Algorithm Configuration", *VISAPP*, 2009, 331-340.

[13] L. Arge, M. Berg., H. Haverkort., & K. Yi, "The priority R-tree: a practically efficient and

worse case optimal R-tree", *in SIGMOD.*, April, 2004, 1-30.
[14] J. Bentley, "Multidimensional binary search trees used for associative searching", *Communication of the ACM, 18*, September, 1975, 509-517.
[15] A. Guttman, "R-trees: A dynamic index structurefor special searching", *SIGMOD, 14*(2), June, 1984, 47-57.
[16] J. H. Friedman, J. L. Bentley and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time", *ACM. TOMS, 3*(3), 1977, 209-226.
[17] C. Silpa-Anan, and R. Hartley, "Optimised KD-trees for fast image descriptor matching", *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, 1-8.
[18] A. Beygelzimer, S. Kakade, and J. Langford, "Cover Trees for Nearest Neighbour", *In Proceedings of 23rd International Conference on Machine Learning*, 2006, 97-104.
[19] Y. Gao, B. Zheng, G. Chen, Q. Li, X. Guao, "Continuous visible nearest neighbour query processing in spatial databases", *Very Large Data Base, 20*(3), 2011, 371-396.
[20] A. Andoni & P. Indyk., "Near-optimal hashing algorithms for approximate nearest neighbour in high dimensions", *Communication of ACM, 51*(1), 2008, 117-122.
[21] J. Brandt, "Transform coding for fast approximate nearest neighbour search in high dimensions", *In IEEE conference on computer vision and pattern recognition*, 2010, 1805-1822.
[22] C. Strecha, A. M. Bronstein, M. M. Bronatein, and P. Fua, "Ldahash: Improved matching with smaller descriptors", *TPAMI, 34*(1), January, 2012, 66-76.
[23] W. Liu, J. Wang, R. J. Y-G. Jang, and S-F Chang., "Supervised hashing with kernels. *In computer vision and pattern recognition*. 2012, 2074-3081
[24] A. Jolly & O. Buisson. (2011). Random maximum margin hashing. in proc. of *Comp. Vis. Pat. Recog*, 2011, 873-880.
[25] P. Indyk and R. Motwani, "Approximate nearest neighbours towards removing the cures of dimensionality", *In STOCK*, 1998, 604-613.
[26] M. S. Charkar, "Similarity estimation techniques from rounding algorithms", *In Proceedings of Annual ACM Symposium on Theory of Computation*, 2002, 380-388.
[2] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality sensitive hashing schemebased on p-stable distribution", *In Proceedings of 2oth Annual Symposium on Computational Geometry*, 2004, 253-262.
[28] H. Jae-Pil, L. Youngwoon, H. Junfeng, C. Shih-Fu, Y. Sung_Eui, "Spherical Hashing: Binary Code Embedding with Hyperspheres", *IEEE transaction on Pattern Analysis and Machine Intelligent*, 2015, 1-14.
[29] J. He, R. Rhadhakrishnan, S-F Chang and C. Bauer, "Compact hashing with joint optimisation of search accuracy and time", *CVPR*. 2011, 753-760.
[30] Y. Gong and S. Lazebnik, "Itetrative Quantisation: a procrustean approach to learning binary codes for large-scale image retrieval", *IEEE transaction on pattern analysis and machine inteliigence*, 35(12), December, 2013, 2916-2929.
[31] A. Torralba, R. fergus, and Y. Weiss, (2008). Small codes and large image databases for recognition.*CVPR*, doi:10.1109/cvpr.2008.4587841
[32] Y. Weiss, A. Torralba, and R. fergus, "Spectral Hashing", *in proceedings of Neural Information Processing System*, 2008, 1-8.
[33] O. Chum, J. Philbin, and A. Zisseman, "Near duplicate image detectionmin-hash and tf-idf weighting", *BMVC*, doi:10.5244/C.22.50
[34] P Jain, B. kulia, K. Grauman, (2008). Fast image search for learned metrics. *CVPR*. doi:10.1109/cvpr.2008.4587841.
[35] M. Rangisky and S. Lazebnik, "Locality sensitive binary codes from shift-invariant kernels", *in proceedings od NIPS*, 2009, 1509-1517.
[36] R. Salakhutdinov and G. Hinton, "Semantic Hashing", *International Journal of Approximate reasoning*, July, 2009, 969-978.
[37] B. Souley and A. U. Othman, "Geometric Similarity Preserving Embedding based Hashing for big data in Clou Computing", *International Journal of Research and Scientific Innovation*, 2019, 10.
[38] J. Wang, S. Kumar and S-F Chang, "Sequential projection learning for hashing with compact codes", in proc. 27th Int. *Conf. Mach. Learn (ICML), 2010,* 1127-1134.
[39] L. Pauleve, H. Jegou and L. Amsaleg, "Locality sensitive Hashing: A comparison of hash function types and queryong mechanism", *Pattern recognition Letters*, 31(11), August, 2010, 1348-1358.
[40] J. Wang, S. Kumar, and S-F Chang, "Semi-supervised hashing for scalable image retrieval", *CVPR*, 2010, 3424-3431.

[41] R-S Lin, D. Rose and J. Yangik, "Spec Hashing: Similarity preserving algorithm for entropy-base coding", *VPR*, June, 2010, 848-854.

[42] R. Ye, and Z. Li, "Compact structure hashing via sparse and similarity preserving embedding", *IEEE transaction on cybernatics, 46*(3), 2016, 718-729.

[43] H. Zhang, L. Liu, Y. Yong, and L. Shao, "Unsupervised deep hashing with pseudo labels for scalable image retrieeval", IEEE transaction on image processing, April, 2018, 1626-1638.

[44] Y. Lv, W. Y. Ng Wing, Z. Zeng, S. D. Yeung,and P. K. Patrick, "Asymmetric Cyslical Hashing for Large Scale Image Retrieval", *IEEE transaction on multimedia, 17*(8), 2015, 1225-1235.

[45] M. Norouzi and D. J. Fleet. (2011). Minimal Hashing for Compact binary codes. *ICML*.

[46] B. Kulis, K. Grauman, "Kernelised Locality-sensitive hashing for scalable image search", *In Proceedings of IEEE conference on computer vision and pattern recognition*, 2009, 2130-2137.

[48] A. Gordo, F. Perronmini, Y. Gong, and S. Lazebnik, "Asymmetric distances for binary embedding", *IEEE Transaction on Pattern Analysis and Machine Intelligence, 36*-(1), January, 2014, 33-47.

[49] B. Kulis and T. Daniel, "Learning to hash with binary reconstructive embeddings", *In Proceedings of NIPS*, 2009, 1042-1050.

[50] C. Yan, H. Xie, D. Yang, J. Yin, and Y. Zhang, "Supervised hash coding with deep neural network for environment perception of intelligent vehicles", *IEEE transaction on intelligent transportation systems, 19*(1), 2018, 284-295.

[51] W. X-M. G. Irie, Z. Li, S-F. Chang, "Locality Linear Hashing for Extracting Non-linear Manifold", *in CVPR*, 2014, 2115-2122.

[52] W. Liu, J. Wang, S. Kumar and S. Chang, (2011). Hashing with graphs. *ICML*.

53] J. Shao, F. Wu. C. Ouyang and X. Zhang, "Sparse spectral hashing", *Pattern Recognition Letters, 33*(3), 2012, 271-277.

[54] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search", *in Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval,* July, 2010, 18-25.

[55] K. He, F. Wen, and I. Sun, "K-means Hashing: An affinity-preserving quantisation method for learning binary compact codes", *in IEEE conference on computer vision and pattern recognition*, June, 2013, 2938-2945.

[56] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit Scalable Deep Hashing with Regularised Similarity Learning for Image retrieval and Person Re-identification", *IEEE transaction on image processing*, August, 2015, 1-14.

[57] W. Dong, M. Charikar, and K. Li, "Asymetric distance estimation with sketchess for similarity search in high-dimensional spaces. *in proceedings of 31st annual international ACM SIGIR conf. Res. Develop. inf. retrieval*, 2008, 123-130.

[58] A. Gordo and F.perronmin. "Asymmetric distances for binary embeddings", *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June, 2011, 729-736.

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

[60] X-J, Wang, I. Zhang, F. Jing, and W-Y. Ma, (2006). Annosearch: Image auto-annotation by search. *In EEE computer society conference on computer vision and pattern recognition*. doi:10:1109/CVPR 2006.58

[61] H. Jagou, R. Tavenard, M. Douze and L. Amsaleg, "Set SIFT 1B", *IEEE Trans. Int. Conf. Acoustic, Speech and Signal Processing*. May, 2011, 861-864.

.