World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:8, No:8, 2014

SC-LSH: An Efficient Indexing Method for Approximate Similarity Search in High Dimensional Space

Sanaa Chafik, ImaneDaoudi, Mounim A. El Yacoubi, Hamid El Ouardi

***Abstract*—**Locality Sensitive Hashing (LSH) is one of the most promising techniques for solving nearest neighbour search problem in high dimensional space. Euclidean LSH is the most popular variation of LSH that has been successfully applied in many multimedia applications. However, the Euclidean LSH presents limitations that affect structure and query performances. The main limitation of the Euclidean LSH is the large memory consumption. In order to achieve a good accuracy, a large number of hash tables is required. In this paper, we propose a new hashing algorithm to overcome the storage space problem and improve query time, while keeping a good accuracy as similar to that achieved by the original Euclidean LSH. The Experimental results on a real large-scale dataset show that the proposed approach achieves good performances and consumes less memory than the Euclidean LSH.

***Keywords*—**Approximate Nearest Neighbor Search, Content based image retrieval (CBIR), Curse of dimensionality, Locality sensitive hashing, Multidimensional indexing, Scalability.

I. Introduction

AS a consequence of the evolution of digital technology and the proliferation of the Internet, multimedia data invade today's world. In spite of the rapid growth of computer performance, it is difficult to perform quick searches in the huge multimedia databases. This problem poses a significant challenge in terms of scalability to many computer vision applications such as image and video database retrieval. Recent research attentions have been shifted to developing efficient and scalable indexing methods that perform management and organization of the large multimedia databases. These methods rely on indexing and structuring tools known as multidimensional indexing. They have been proposed to solve the nearest neighbors (*NN*) search problem, which is consist in finding closest (or most similar) data points to a given query in high dimensional space.

Multidimensional indexing methods are organized into two main families: the family of conventional indexing methods and the family of approximate indexing methods. The first one is based on indexing data points according to their distribution and location in the multidimensional space. A number of efficient conventional algorithms (e.g. R-tree [1] and KD-tree [2]) have been proposed for the *NN* search problem. Unfortunately, this family of methods suffers from scalability issues, as query time is exponential in d. In fact, for large enough d ($d > 16$), the performances of this family get significantly degraded and become worse than the full sequential scan approach that compares a query to every point from the dataset. This problem is known as "The dimensionality curse"[3].

The second family has been proposed to overcome the dimensionality curse problem by using the approximation approach. This family is based on data compression allowing for a sequential scan of a small approximation file instead of reading the complete data file. Thus, it provides slightly less accurate results, but offers in return a sharp reduction in computation time. We find in this family VA-File [4] and its variants [5], [6], the LSH methods [15], [17] and its variants [7]-[13], [14], BitMatrix [10], KRA+-Block [11], multi-dimensional inverted index [22], etc.

Unluckily, most of the indexing and search structures available today have several limitations including scalability: 1) a relatively large computation time 2) limiting memory 3) degradation of research quality.

We focus in this paper on one of the most popular methods for performing approximate search in high dimensional space based on the concept of locality-sensitive hashing (LSH). The choice of this class of methods is justified by their quality and effectiveness with respect to several multidimensional indexing methods [12], [20]. Furthermore, it has been successfully applied to various domains, including computer vision, web clustering [21] and computational biology [23].

Although LSH is a promising approach for solving the *NN* search problem, it has limitations that affect structure and query performance. The main limitation of the basic LSH is memory consumption. In order to increase the probability to map similar points to similar hash values, the LSH concatenates several random independent hash functions, which leads to a severe redundancy of the hash values as well as a redundancy of the hash tables. Moreover, when using large scale dataset, more hash tables are required leading to high memory consumption.

In this paper, we propose a new LSH-based approach called Symmetries of a cube LSH (SC-LSH) for efficient high dimensional *NN* search. The new LSH scheme allows the concatenation of random dependent hash function, which reduces the hash values redundancy and increases the collision probability. Moreover, the SC-LSH saves much storage space and speeds up the search process, while keeping a good accuracy that is similar to the original Euclidean LSH one.

Sanaa Chafik, Imane Daoudi, and Hamid El Ouardi are with the Computer Science, Systems and Renewable Energy Lab, Hassan II Ain Chock University, National School of Electrical and Mechanical (ENSEM), Casablanca, Morocco (phone: 212-673-619-257; e-mail: sanaa.chafik89@gmail.com, {i.daoudi, h.elouardi}@ensem.ac.ma).

Mounim A. El Yacoubi is with Intermedia Lab, Telecom SudParis / Mines-Telecom Institute, Paris - France (e-mail: mounim.el_yacoubi@telecom-sudparis.eu).

The rest of this paper is organized as follows. We first discuss preliminaries in Section II, and then introduce our algorithm in Section III. Section IV gives the experimental results that compare our algorithm with the Euclidean LSH. Finally, conclusions and perspectives are presented in Section V.

II. Preliminaries

In this section, we give a brief overview of LSH and the main LSH functions families discovered for different similarity measures. Then, a Euclidean LSH method is presented.

TABLE I
NOTATIONS

d	Dimension
n	Number of points of the dataset
$d(.,.)$	Distance
$h_{i,j}$	Hash function
NN	Nearest Neighbor
K	Number of Hash functions
L	Number of Hash tables
$B(p,r)$	Ball of radius r centered at a data point p
$\|.,.\|$	The Euclidean Distance
$P_r[.]$	Probability
f_p	Probability density function
$\Re$	Set of real numbers
U	Universe
Z	Set of integers
$O(.)$	Complexity

A. Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) is one of the most popular methods for solving the *NN* search problem in high dimensional space. The LSH is based on the use of several hash functions describing the distribution of data in multidimensional space. The hash function assigns the same hash value to the closest points with high probability. Then, the *NN*s are performed by hashing the query point and retrieving points stored in same buckets[1] of the hash table to the query. The LSH approach relies on locality-sensitive hash functions. A family F of hash functions is called a locality Sensitive family if it satisfies the following definitions:

Definition.1 [15]. *An LSH family* $F = \{\Re^d \to U\}$ *is called* (r_1, r_2, P_1, P_2) *-sensitive, if for any* $p,q \in \Re^d$, *with* $P_1 > P_2$ *and* $r_1 < r_2$ *:*

- If $p \in B(q, r_1)$ then $P_{rF}[h(p) = h(q)] \geq P_1$
- If $p \notin B(q, r_2)$ then $P_{rF}[h(p) = h(q)] \leq P_2$

[1]Buckets: Data type that groups objects together, the term is used when discussing hashing algorithms, where different items that have the same hash code (hash value) go into the same "bucket".

Definition.2 [16]. *A locality sensitive hashing scheme is a distribution on a family* F *of hash functions operating on a collection of points, such that for two points* p,q

$$P_{rF}[h(p) = h(q)] = sim(p,q)$$

Here $sim(p,q)$ *is some similarity function defined on the collection of points.*

To increase the gap between high probability P_1 and low probability P_2, and to achieve high search accuracy, it is necessary to use several hash functions defined as: $g_j : \Re^d \longrightarrow Z^K$, $g_j(q) = (h_{1,j}(q),...,h_{K,j}(q))$, characterizing the hash tables, where $h_{i,j}(1 \leq i \leq K, 1 \leq j \leq L)$ are selected randomly and independently from F. Those functions hash all data points and build the index structure, by placing each point p in a bucket $g_j(p)$. It is worth noting that the buckets store only the references of points to facilitate access to data points. The search algorithm for a query point q begins by computing the hash values $g_1(q),...,g_L(q)$, then the distance between the query point and the points of the same bucket. Finally, the nearest points to the query are selected; they represent the results of the research.

B. LSH Families

In the last few years, the development of locality sensitive hash functions has been well addressed in the literature. We next briefly survey the main LSH functions families and their properties.

Min-hash [19]: It is a family of hash functions allowing for the estimation of the similarity between two sets. It consists of selecting a random permutation π from the space U. For a set of data A, the hash function is defined as: $h(A) = \min\{\pi(a) \mid a \in A\}$. The probability of collision between two sets A and B is defined as: $P_r[h(A) = h(B)] = s(A,B)$, where $s(A,B)$ is the Jaccard coefficient defined as $s(A,B) = \frac{|A \cap B|}{|A \cup B|}$. This technique is often used by search engines to detect redundant web pages; it has also been applied in large-scale clustering problems.

LSH based on Hamming distance [15]: It is a family of hash functions adapted to the Hamming distance. For d-dimensional vectors $\{0,1\}^d$, the family F of hash functions is the set of projections on one of the d coordinates: $F = \{h : \{0,1\}^d \longrightarrow \{0,1\} \mid h(x) = x_i, i = 1,...,d\}$, where x_i is i^{th} coordinate of x. The function h is randomly selected from F, and allows choosing a random bit vector.

LSH based on cosine distance [16]: is a LSH family gathering the following hash function:

$$h_r(\vec{p}) = \begin{cases} 0 \ if \ \vec{p}.\vec{r} < 0 \\ 1 \ if \ \vec{p}.\vec{r} \geq 0 \end{cases}$$

These functions allow for partitioning the space into two spaces by a randomly chosen hyperplane defined by the unit normal vector $\vec{r}$. The probability of collision between two vectors $\vec{p}$ and $\vec{q}$ is calculated as follows:

$$P_r[h(\vec{p}) = h(\vec{q})] = 1 - \frac{\theta(\vec{p},\vec{q})}{\pi}$$

LSH based on l_p-distance [17]: It is an interesting LSH family gathering the following hash functions:

$$h_{a,b}(p) = \left\lfloor \frac{a.p + b}{w} \right\rfloor$$

where a is a random d-dimensional vector of p-stable distribution, b is a real number chosen uniformly from $[0, w]$ and w is a user-specified constant.

The probability of collision between two points p and q is defined as:

$$p(c) = P_r[h(\vec{p}) = h(\vec{q})] = \int_0^w \frac{1}{c} f_p\left(\frac{t}{c}\right)\left(1 - \frac{t}{w}\right)dt$$

This family is defined for the case where the distances are measured according to the l_p metric, for any $p \in [1,2]$.

This LSH family provides an efficient solution to randomize the NN search problem in high dimensional spaces [17].

LSH based on χ^2-distance [7]: It is a new LSH family fitted to the χ^2-distance, defined by the following hash function:

$$h_{a,b}(p) = \left\lfloor \frac{\sqrt{\left(8.\frac{a.p}{w^2} + 1\right)} - 1}{2} + b \right\rfloor$$

where a is a random d-dimensional vector, b is a real number chosen uniformly from $[0,1[$ and w is a scalar for the quantization width, calculated using χ^2-distance [7].

This LSH family was successfully applied in the context of image and video retrieval when data are represented by histograms.

In the following, we will describe one of the most famous LSH technique for solving *NN* search problem, it based on l_p-distance and it called Euclidean Exact LSH (E2LSH).

C. The E2LSH Method

The Euclidean Locality-Sensitive Hashing (E2LSH) method was proposed in [18] as a solution for the high dimensional $(R, 1-\delta)$-NN search problem in the Euclidean space l_2. For a query point q, the E2LSH method reports all points $p \in \Re^d$ satisfying $\|p - q\|_2 \le R$ with a probability at least equal to $1-\delta$ (δ is the probability that a near neighbor p is not reported).

The index structure of the E2LSH is performed by calculating the hash value of each point of the dataset, using the hash function $h_{a,b}(p): \Re^d \longrightarrow Z$, defined as

$$h_{a,b}(p) = \left\lfloor \frac{a.p + b}{w} \right\rfloor$$

where a is a random d-dimensional vector of p-stable distribution [17] and b is a real number chosen uniformly from $[0, w]$, where w is a positive real number. Intuitively, the hash function $h_{a,b}(p)$ projects the point p on a line whose direction is identified by the d-dimensional vector a. Then, the projection of the point p is shifted by a constant. In such a scheme, the line is segmented into intervals of length w.

To improve the hashing discriminative power, a second hash function g_j is constructed using several functions $h_{i,j}$ defined as: $g_j(p) = (h_{1,j}(p), \ldots h_{K,j}(p))$, where $(1 \le j \le L)$. The similar points of the data set (having the same $g_j(p)$) are stored in the same bucket of the hash tables. We note that the E2LSH performances depend crucially on parameters K and L: K refers to the number of hash functions and L represents the number of hash tables. The value of L is determined by requiring that the probability to find the true *NN*s is at least $1-\delta$, which implies:

$$L = \left\lfloor \frac{\log \delta}{\log(1 - P_1^K)} \right\rfloor$$

The K value is chosen as a function of the dataset to minimize the query time.

Although the E2LSH is, a promising approach for solving the *NN* search problem, but present limitation in memory space storage. To insure good performances the E2LSH requires the use of several hash tables, which requires too much memory space. The next section presents a new LSH scheme to overcome the E2LSH storage problem.

III. The Proposed Approach

In this section, we present the index structure and the nearest neighbor search algorithm of the new proposed approach SC-LSH.

A. The Index Structure

We propose a new LSH-based approach called Symmetries of a cube Locality Sensitive Hashing (SC-LSH) for efficient memory space management. The basic idea of the new approach is structuring the high dimensional space into a set of disjoints cubes. For this purpose, the SC-LSH propose to

project dataset points on symmetry axes of a cube, (Fig. 1), using the hash function $h_{a,b}(v):\Re^d \longrightarrow Z$, defined as

$$h_{a,b}(v)=\left\lfloor \frac{a.v+b}{w} \right\rfloor$$

where a is a d -dimensional vector representing the symmetry axes of a cube, and b is a real number chosen uniformly from $[0,w]$, where w is a positive real number depending on the length of the edge of a cube e. For the space diagonal, the value of w is equal to $e\sqrt{3}$ and for the axes of rotation, the value of w is equal to e (Fig. 1).

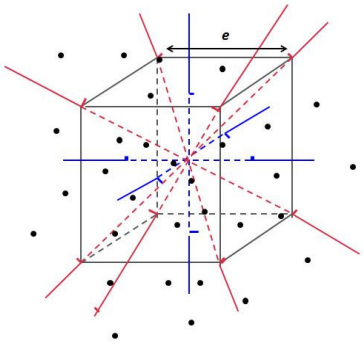

Fig. 1 Symmetries of a cube hashing

After an experimental analysis on the approach performances, we restricted the number of symmetry axes to seven. The first one is drawn independently from Gaussian distribution and the others are generated as described above, respecting the mathematical properties of the symmetries of the cube. Therefore, this process enables to hash similar points to same cube of edge length e defined as $g(p)=(h_1(p),...h_7(p))$, (Fig. 2). To ensure a good accuracy it is necessary to use multiple hash tables, an experimental study was proposed to choose the appropriate number of hash table.

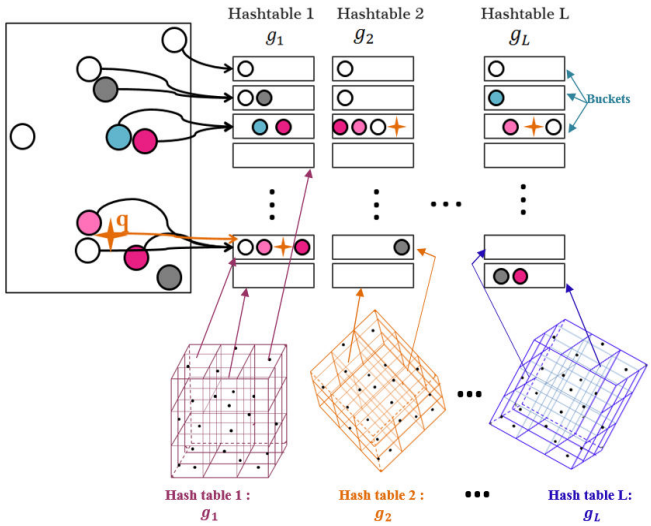

Fig. 2 The index structure of the SC-LSH

B. NN Search Algorithm

Like in the E2LSH approach, the similarity search process begins in our new approach by computing the hash values of a query point q, $g_j(q)=(h_{1,j}(q),...h_{7,j}(q))$ with $(1\le j\le L)$, which index the query point with the appropriate buckets. Then the Euclidean distance between the query and the vectors of the same or the nearest bucket is computed and the closest points to the query are returned.

IV. Experimental Result

In this section, we evaluate the proposed approach on the high dimensional *NN* search problem using real and synthetic datasets. We compared the SC-LSH performances with the E2LSH approach.

A. Datasets and Evaluations Metrics

We perform the experiments with two large-scale datasets:

*SIFT1*M [24]: it contains one million SIFT descriptors extracted from random images; each descriptor is represented by a 128-dimensions point.

SynGIST: synthetic datasets generated from *GIST1M* [24]

We randomly choose 100 data points to form a query set and use the remaining ones to form the gallery database.

We use the same criterion as in [13] to evaluate the query accuracy. For a given query q, let $p_1,p_2,...,p_m$ be the m -*NN*s retrieved points, sorted in ascending order of their distances to q. Let $p_1^*,p_2^*,...,p_m^*$ be the true m -*NN*s, we define the *rank-i approximation ratio* as:

$$R_i(q)=\frac{\|p_i,q\|}{\|p_i^*,q\|}$$

The overall ratio is defined as

$$\frac{1}{m}\sum_{i=1}^{m}R_i(q)$$

The more closely the overall ratio to one, the more accurate the results are, and when it equals to one, the results are exact.

For a set of queries Q, the average overall ratio is defined, reflecting the general quality of all $m-NNs$. In order to scrutinize the quality of neighbors at individual ranks the *average rank-i ratio* is defined, which is the mean of the *rank-i approximation ratio* of all queries in Q, namely, defined as:

$$\frac{1}{|Q|}\sum_{\forall q\in Q}R_i(q)$$

In this paper, we use the *average rank-i ratio* and the *mean average rank-i ratio* (MARR) of all m -*NN*s retrieved points, to evaluate similarity search quality.

In addition, we also report the mean query time and the space consumption (size of the index structure $O(nL)$).

Our tests were conducted on a Linux distribution: Ubuntu 12.04 machine 1.86GHz with 6GB RAM and 146 GB of local disk.

B. Parameter Settings

The main goal of selecting optimal parameters is to enable high quality search with high speed, while using a small amount of memory. The proposed approach depends mainly on the number of hash tables L , which defines the accuracy: a large number of hash tables increases the collision probability. Therefore, it affects the memory space storage.

The SC-LSH approach depends also on the length of the edge e , that defines the volume of the cube and the number ofpoints indexed inside. A large edge increases the probability of collision but increases the computation time.

To choose properly the optimal edge size e we perform several tests on different edge sizes. We noticed that the value $e = 4$ provides good accuracy and fast query time. To find the optimal number of hash tables L , we perform several tests for different values of L on *SIFT1M* dataset. Table II summarizes the performances of the SC-LSH method for different values of L .

TABLE II
PERFORMANCE OF THE SC-LSH FOR DIFFERENT VALUE OF L ON *SIFT1M* DATABASE

L	10	20	30	40	50	60
MARR	0.725	0.741	0.747	0.749	0.750	0.750
Query Time (s)	0.1062	0.1536	0.1842	0.2086	0.2189	0.2358
Memory Consumption (Gb)	0.1117	0.2235	0.3352	0.4470	0.5587	0.6705

In the following, we set manually the value of the edge of the cube at $e = 4$, and we chose $L = 30$ as the appropriate number of hash table, for indexing the S*IFT1M* dataset.

We performed the same performance tests on synthetic datasets *SynGIST* to define the suitable number of hash tables L .

C. Performances on Real Dataset

In this section, we report the evaluation results of the proposed approach using *SIFT1M*, we compared the result obtained with the E2LSH method.

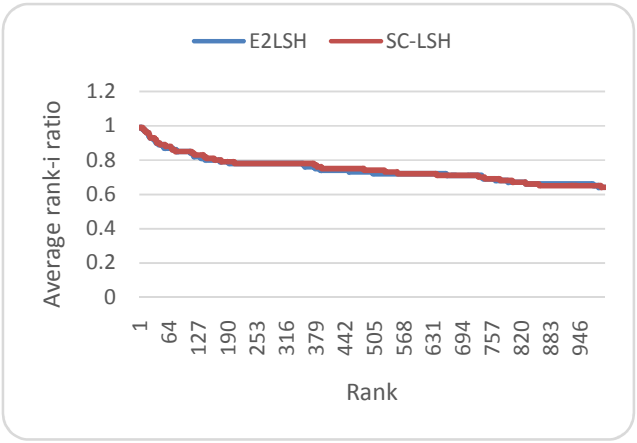

Fig. 3 Average rank-i ration on *SIFT1M* dataset of 1000-*NNs*

Fig. 3 shows the search quality test of the two approaches SC-LSH and E2LSH performed on *SIFT1M*dataset. We note that both methods provide almost the same similarity search quality. The proposed approach outperforms slightly the E2LSH method. In fact, the *mean average rank-i ratio* of the SC-LSH is equal to 0,747, while the *mean average rank-i ratio* of the Euclidean LSH is equal to 0,744. Therefore, the proposed approach requires less computation time and memory space compared to the E2LSH approach (see Table III).

TABLE III
PERFORMANCE OF THE SC-LSH AND E2LSH ON SIFT1M DATABASE

SC-LSH	Query Time (s)	0.1842
	Memory Consumption (GB)	0.3352
E2LSH	Query Time (s)	1.0343
	Memory Consumption (GB)	4.2244

D. Performances on Synthetic Dataset

To evaluate the behavior of the SC-LSH in scaling, we performed several tests on different synthetic databases of different sizes and dimensionalities.

1. Search Quality Test

Tables IV and V show the search quality of the proposed approach and the Euclidean LSH with respect to a dataset of size n and dimensionality d . As the dimensionality d increases, the search quality of the two approaches decreases. We note that the SC-LSH and the E2LSH underperform in terms of quality of search for large dimensions, but keep a good accuracy for large dataset sizes.

TABLE IV
MEAN AVERAGE RANK-*I* RATIO OF 1000-NNS VS. DIMENSIONALITY FOR $n = 250000$

Dimension	100	300	500	700
SC-LSH	0.990713	0.514426	0.172545	0.06109
E2LSH	0.990120	0.514006	0.170722	0.06053

TABLE V
MEAN AVERAGE RANK-*I* RATIO OF 1000-NNS VS. DATASET SIZE FOR $d=100$

Database size	250000	500000	1000000
SC-LSH	0.990713	0.980763	0.963412
E2LSH	0.990120	0.98	0.963245

2. Query Time Test

To evaluate the timing in large scale of the proposed approach, we computed the query time of the latter for different dimensionalities and database sizes.

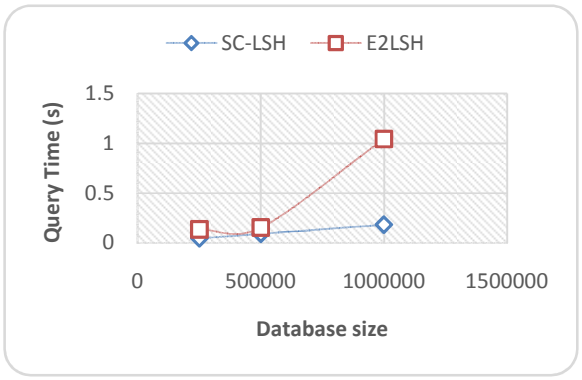

Fig. 4 Query time vs. database size for $d=100$

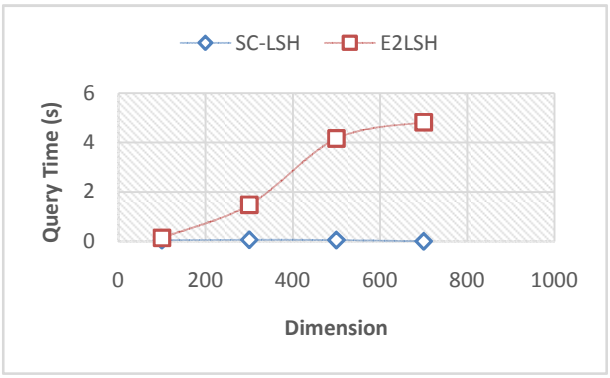

Fig. 5 Query time vs. dimensionality for $n=250000$

Figs. 4 and 5 show the speedup of the SC-LSH and the E2LSH with respect to database size and dimensionality. We can observe that the proposed approach outperforms the Euclidean LSH in timing: the SC-LSH is four times faster than the E2LSH. This is due to the new structuring method that hashes similar points in the same cube and thus speeds up the search process. We note that the proposed approach allows having the same search quality with a reduced query time.

3. Memory Consumption Test

The Locality Sensitive Hashing scheme is based on the use of several hash tables storing the similar points in the appropriate bucket, which requires a large amount of memory. Table VI and VII show the space required for storing the hash tables of the SC-LSH and the E2LSH approaches for different database sizes and dimensions. We observe that the space consumed by the proposed approach remains small and shows a slight increase when the dataset size and dimension increase. On the other side, the Euclidean LSH requires more memory space, more than four times the space required by the proposed approach. Moreover, the Euclidean LSH keeps the same number of hash tables for different dimensions (for $n=250000$we find $L=1485$ equivalent to 4.149 GB for different dimensionalities).

TABLE VI
MEMORY CONSUMPTION VS. DIMENSIONALITY FOR $n=250000$

Dimension	100	300	500	700
SC-LSH	0.0279	0.0838	0.1117	0.1396

TABLE VII
MEMORY CONSUMPTION VS. DATASET SIZE FOR $d=100$

Database size	250000	500000	1000000
SC-LSH	0.0279	0.0558	0.1117
E2LSH	4.1490	3.3248	4.2244

V. CONCLUSION

In this paper, we presented a new approach based on the projection on symmetries of the cube. This projection allows structuring and gathering similar points in same cube.

We performed several tests on different databases to evaluate the behavior of the proposed approach on scaling. In terms of computational time and memory cost, we compared the result obtained with the Euclidean LSH. We notice that the proposed approach outperforms the original Euclidean LSH in computational time and memory cost for large dataset. In our future work based on the proposed SC-LSH approach, we intend to further improve the search accuracy especially in high dimensional datasets.

ACKNOWLEDGMENT

We sincerely thank A. Andoni for providing us the E2LSH package.

REFERENCES

[1] A. Guttman. "R-trees: A dynamic index structure for spatial searching". In SIGMOD, 1984, pp. 47-57.
[2] J. L. Bentley. "K-d trees for semidynamic point sets". InSoCG, 1990, pp. 187-197.
[3] Weber, H.J. Schek and S. Blott, "A quantitative analysis and performance, study for similarity-search methods in high-dimensional spaces". In Proceedings of 24rd International Conference on Very Large Data Bases, New York, USA, 1998, pp.194-205.
[4] S. Blott and R. Weber."A Simple Vector-Approximation File for Similarity Search in High-Dimensional Vector Spaces". Technical Report 19, ESPRIT project HERMES, March 1997.
[5] L. Ye and Y. Hua. "The CMVAI-File: An Efficient Approximation-Based High-Dimensional Index Structure",Multimedia Information Networking and Security (MINES), 2010.
[6] H.Lu, B.C.Ooi, H.T.Shen and X.Xue. "Hierarchical Indexing Structure for Efficient Similarity Search in Video Retrieval". IEEE Transactions on Knowledge and Data Engineering, November 2006.
[7] D. Gorisse, M. Cord and F. Precioso. "Locality-sensitive hashing for chi2-Distance", IEEE Transactions on Pattern Analysis and Machine Intelligence 34, 2012, pp.402–409.
[8] H. Wang, J. Cao, L. Shu, and D. Rafiei."Locality sensitive hashing revisited: Filling the gap between theory and algorithm analysis". In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, 2013, pp. 1969-1978.

[9] A. Andoni and P. Indyk. "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions". In 47th Annual IEEE Symposium on foundations of Computer Science, 2006, pp. 459-468.
[10] C. Calistru, C. Ribeiro and G.David. "High- Dimensional Indexing for Video Retrieval". Multimedia - A Multidisciplinary Approach to Complex Issues, 2012.
[11] I. Daoudi, K. Idrissi and S.E.A Ouatik. "Kernel region approximation blocks for indexing heterogonous databases". Multimedia and Expo, 2008 IEEE International Conference, 2008, pp. 1237-1240.
[12] T. Liu, A. W. Moore, A. G. Gray and K. Yang. "An Investigation of Practical Approximate Nearest Neighbor Algorithm". In proceeding of: Advances in Neural Information Processing Systems 17. Vancouver, British Columbia, Canada, 2004.
[13] J. Gan, J. Feng, Q. Fang and W.Ng, "Locality-sensitive hashing scheme based on dynamic collision counting". In proceeding SIGMOD '12 Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. New York, USA, 2012, pp. 541-552.
[14] Q. Lv, W. Josephson, Z. Wang, M. Charikar and K.Li. "Multi-probe LSH: efficient indexing for high-dimensional similarity search". In VLDB, 2007, pp.950-96.
[15] P.Indyk, and R.Motwani, "Approximate nearest neighbor: Towards removing the curse of dimensionality". In Proceeding STOC '98 Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998, pp.604-613.
[16] M.S. Charikar. "Similarity estimation techniques from rounding algorithms". In Proceedings of the Symposium on Theory of Computing, 2002.
[17] M. Datar, N. Immorlica, P. Indyk and V. Mirrokni. "Locality sensitive hashing scheme based on p-stable distributions". In Proceedings of the ACM Symposium on Computational Geometry, 2004.
[18] A. Andoni and P. Indyk. E2LSH: Exact Euclidean locality-sensitive hashing. Implementationavailableat:http://www.mit.edu/~andoni/LSH/.
[19] A. Z. Broder, M. Charikar, A. M. Frieze, andM. Mitzenmacher. "Min-wise independent permutations". In STOC, 1998, pp. 327–336.
[20] S. Chafik, I.Daoudi, M.A EL Yacoubi, H. El Ouardi and B. Dorizzi. "Locality sensitive hashing for content-based image retrieval: A comparative experimental study". The Fifth International Conference on Next Generation Networks and services (NGNs), 2014.(unpublished)
[21] Haveliwala, T., Gionis, A. and Indyk. "Scalable Techniques for Clustering the Web" (Extended Abstract). In Third International Workshop on the Web and Databases (WebDB 2000), Dallas, Texas, 2000.
[22] D. Feng and J. Yang, C. Liu. "An efficient indexing method for content-based image retrieval".Neurocomputing, April 2013, pp.103-114.
[23] J. Buhler and M. Tompa. "Finding motifs using randomprojections". In Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB1), 2002, pp. 225-242.
[24] http://corpus-texmex.irisa.fr