This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2018.2872898, IEEE Transactions on Multimedia

equivalent to the weights in dimension, the increase or decrease in value is considered as weighted hamming distance (WHD). Therefore, in the process of increasing the weighted hamming distance for NN search, each kind of dimension has different cases, rather than a simple 0/1 bit flip. So, we will discuss different situations separately. Specific steps are as follows:

1) Resize the *mask*. The number of 1 in *mask* (in binary) equals to the WHD. Further, the number of 1 in a dimension of *mask* represents the WHD of the corresponding dimension. Assume that c_1 is the number of 00 and 11, and c_2 is the number of 10 and 01. As the range of WHD of 11 and 00 is 3, and that of 01 and 10 is 2, the length of the *mask* is set to be $3 \times c_1 + 2 \times c_2$, where $c_1 + c_2 = b/2$. For example, as shown in Table II, the changes of the *mask* 00011011 are presented, where the WHD ranges from 0 to 10.

TABLE II
THE CHANGES OF MASK FOR 8-BIT BINARY CODE 00011011 WHEN INCREASING THE WEIGHTED HAMMING DISTANCE.

WHD \ binary code	00	01	10	11	number of cases
	one of the cases				
r=0	000	00	00	000	1
r=1	000	00	00	001	C_{10}^{1}
r=2	000	01	10	000	C_{10}^{2}
r=3	011	01	00	000	C_{10}^{3}
…	……				
r=10	111	11	11	111	1

2) Transform the *mask*. When the WHD increases, the transformation of *mask* for NN search is the same as MIH except that the length of the *mask* is different. So, when the WHD increases to *R*, the total number of the cases of *mask* is $C_{3\times c1+2\times c2}^{R}$. In each case, there exist different bit-wise flips in *mask*. We first initialize two variables *IN* and *RN* to 0, which represent the numbers to be added and subtracted to the query, respectively. Note that the two variables have the same number of bits as the query, as every two bits of them represents the WHD of the corresponding dimension.

TABLE III
THE CHANGES OF CORRESPONDING DIMENSION OF *IN* AND *RN* WHEN BINARY CODE IS 00 OR 11. '+' REPRESENTS THE CHANGE IN *IN*; '-' REPRESENTS THE CHANGE IN *RN*.

value \ mask	000	001	010	011	100	101	110	111
00	0	+1	+1	+2	+1	+2	+2	+3
11	0	-1	-1	-2	-1	-2	-2	-3

TABLE IV
THE CHANGES OF CORRESPONDING DIMENSION OF *IN* AND *RN* WHEN BINARY CODE IS 01 OR 10. '+' REPRESENTS THE CHANGE IN *IN*; '-' REPRESENTS THE CHANGE IN *RN*.

value \ mask	00	01	10	11
01	0	+1	-1	+2
10	0	+1	-1	-2

Table III and Table IV are presented to detail the changes of *IN* and *RN*, according to different *mask* and binary codes. For 00 (see Table III), only *IN* can be used, meaning a certain number can be added to the dimension. While, 11 is just on the contrary. As for 01 (see Table IV), the change ranges from -1 to $+2$, which indicates *IN* can be set as 01 and 10 or *RN* can be set as 01 in corresponding dimension. However, when the WHD is 1, *IN* and *RN* can both be utilized. In this case, we use *mask* to determine which variable to change. It requires that when the dimension of *mask* equals 01, the corresponding dimension of *IN* is set as 01, while *RN* is set as 01 when *mask* is 10. The dimension of 10 works the same way. With the simple approach, it guarantees that each transformation of *mask* corresponds to only one situation, and takes into account all the cases at the same time.

3) Examine the hash buckets. After the previous steps, the address of the hash bucket to be examined is

$$\text{address} = \text{query} + IN - RN. \tag{9}$$

Subsequent search approaches are consistent with the MIH.

Here is an example, given the query 01001011, which is 8-bit code (4-dimension), and the length of the *mask* is set to be 10-bit long. If the expected WHD is 4, the *mask* should contain four bits valued 1. Assume the *mask* is 0010110100, for the first dimension (from right to left), the corresponding dimension of *mask* does not contain any 1, so there is no change in *IN* or *RN*. For the second dimension of the query 00, the *mask* (containing 3 bits in this dimension) has two bits valued 1, so *IN* = *IN* | 00100000. For the third dimension of the query 10, the corresponding *mask* has one bit valued 1, then $RN = RN \mid 00000100$. For the fourth dimension 11 of the query, there is one bit valued 1, so $RN = RN|00000001$. As a result, the *IN* equals to 00100000 and *RN* equals to00000101. Thus, the address of hash bucket to be examined is 01100110, which has the WHD of 4 for the query.

This method takes into account all cases of the WHD, and each case appears only once. To fit DBQ binary codes, it considers different combinations of *mask* and binary code to get the address of hash buckets to be examined. As revealed in the experiment results, DBIH remarkably increases the search efficiency compared to the linear search.

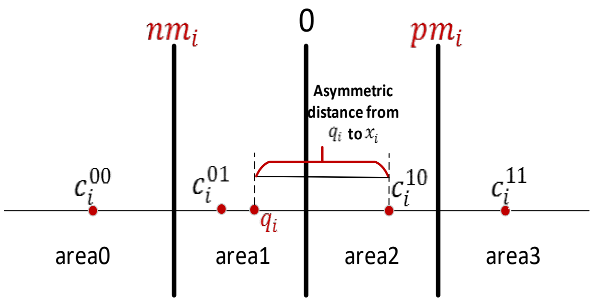

Fig. 4. Asymmetric distance.

D.! Weighted Distance Measurement

Binary codes are storage efficient and fast to compute. Millions of binary codes can be compared to a query in less

On the other hand, weighted hamming distance has more distinguishing ability. When one-bit quantization is used, the hamming distance of each two-bit code has three possible values (0, 1 and 2). But it has four possibilities (0, 1, 2 and 3) when using DBQ. Thus, weighted hamming distance has a wider range than original method. For example, assume the number of bit is 64, and then the range of hamming distance is 0 to 64. However, the range of weighted hamming distance is 0 to 128, which is twice of the former. As a consequence, DBQ obtains stronger discrimination than traditional binary embedding methods.

- Double-bit quantization *VS* Manhattan hashing [48]

To verify the effectiveness of DBQ, Manhattan hashing (MH) is compared. The basic idea of MH is to encode each projected dimension with multiple bits of natural binary code, based on which the Manhattan distance between points in the hamming space is calculated. In order to maintain consistency of the experiments, the number of bit per dimension is set to be 2. The dataset we use is BIGANN SIFT 1M. The experiment has 1000 queries, in which precision and recall are adopted as metrics. As shown in table VIII, DBQ outperforms MH in most of the cases expect when the hashing method is LSH due to the randomness.

Note that NPQ has better performance than MH[50]. To our knowledge, DBQ has the best accuracy on the three datasets at present. For the reason of paper length, wo do not compare our method with NPQ.

2) *Double-bit Index Hashing*

Each experiment involves 10000 queries, and we report the average running time. When the number of bits is 64, we divide the binary codes into 4 segments. The 128-bit long binary codes are divided to 8 segments. The results illustrate that DBIH is remarkably faster compared to the linear scan. Note that the linear scan speed relies heavily on the memory cache. If there is less cache, linear scan will be much slower [32].

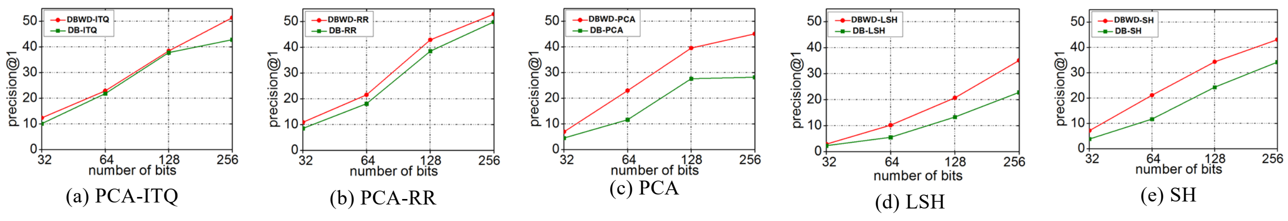

Fig. 8. Influence to precision of the weighted distance measurement on the BIGANN dataset using 128-d sift descriptors through double-bit quantization and different binary embedding methods, including ITQ, RR, PCA, LSH and SH. From left to right: 32-bit, 64-bit, 128-bit and 256-bit codes. DB represents the double-bit quantization and DBWD means the double-bit quantization with weighted distance measurement.

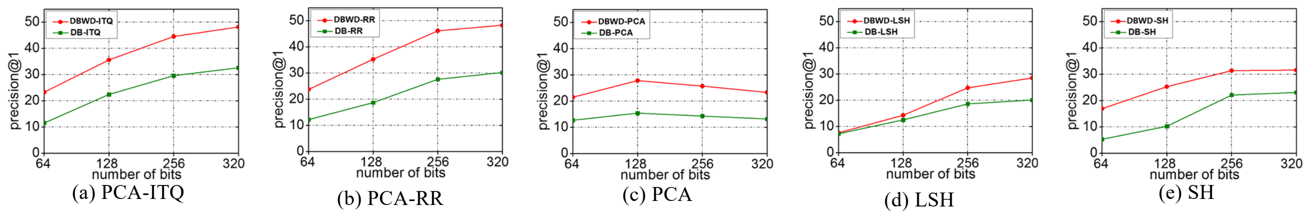

Fig. 9. Influence to precision of the weighted distance measurement on the CalTeach101 dataset using 320-d gist descriptors through double-bit quantization and different binary embedding methods, including ITQ, RR, PCA, LSH and SH. From left to right: 64-bit, 128-bit, 256-bit and 320-bit codes. DB represents the double-bit quantization and DBWD means the double-bit quantization with weighted distance measurement.

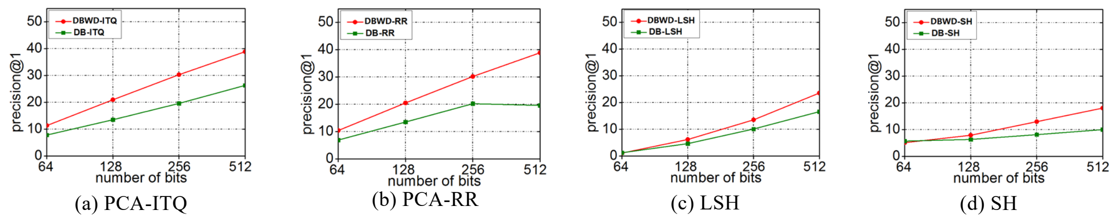

Fig. 10. Influence to precision of the weighted distance measurement on the BIGANN dataset using 960-d gist descriptors through double-bit quantization and different binary embedding methods, including ITQ, RR, LSH and SH. From left to right: 64-bit, 128-bit, 256-bit and 512-bit codes. DB represents the double-bit quantization and DBWD means the double-bit quantization with weighted distance measurement.

DBIH only needs small number of hash buckets to be examined. Fig.5 (a) displays the average search radius required for1000 queries on a dataset of 10^8 SIFT descriptors. As we can see, the search radiuses are concentrated in the range of 2 to 4. Fig.5 (b) shows the relationship between the number of examined hash buckets and the hamming distance on the same dataset. For linear scan, each signature is equivalent to one hash bucket. So it elaborates that DBIH requires a much smaller