GRASSMANN HASHING FOR APPROXIMATE NEAREST NEIGHBOR SEARCH IN HIGH DIMENSIONAL SPACE

Anonymous ICME submission

ABSTRACT

Locality-Sensitive Hashing (LSH) approximates nearest neighbors in high dimensions by projecting original data into low-dimensional subspaces. The basic idea is to hash data samples to ensure that the probability of collision is much higher for samples that are close to each other than for those that are far apart. However, by applying k random hashing functions on original data, LSH fails to find the most discriminant hashing-subspaces, so the nearest neighbor approximation is inefficient. To alleviate this problem, we propose the Grassmann Hashing (GRASH) for approximating nearest neighbors in high dimensions. GRASH first introduces a set of subspace candidates from Linear Discriminant Analysis (LDA); and then it applies Grassmann metric to select the optimal subspaces for hashing; at last it generates hashing codes based on non-uniform bucket size design motivated by Lloyd-Max quantization. The proposed GRASH model enjoys a number of merits: 1) GRASH introduces the Grassmann metric to measure similarity between different hashing subspaces, so the hashing function could better capture data diversity; 2) GRASH obtains the subspace candidates from LDA, so it could incorporate the discriminant information into the hashing functions; 3) GRASH extends LSH's 1-d hashing subspaces to m-d, i.e. it is a multidimensional extension of hashing approximation; 4) motivated by Lloyd-Max quantization, GRASH applies non-uniform size bucket to generate hashing codes, so the distortion can be minimized. Experimental results on a number of datasets confirm the validity of our proposed model.

Index Terms*—*hashing, subspace learning, grassmann manifold, optimization

1. INTRODUCTION

Similarity search is a common task in many applications. It involves a collection of samples which are characterized by a set of features and represented as points in a multi-dimensional space [1][10]. The K Nearest Neighbors (K-NN) serves as a solution to similarity search. When the number of dimensions d is small, K-NN solves this problem well; however, when d is large, conventional indexing solutions for K-NN search is known to suffer from the "curse of dimensionality" [1][4][5][6], i.e., for large enough d, it provides litter little improvement over a linear search. Therefore, K-NN poses an obstacle to the further development of large scale similarity search system.

To solve this problem, Locality-Sensitive Hashing (LSH) [1][6] is proposed. LSH approximates similarity search based on hashing. The basic idea is to hash samples in high dimensions so as to ensure the probability of collision is much higher for objects that are close to each other than for those that are far apart. However, as original proposed, LSH suffers from a number of disadvantages. 1) LSH applies random hashing functions on original data, so it fails to incorporate the distances between different hashing subspaces; moreover, LSH doesn't incorporate the discriminant information between samples, i.e. there is no penalty for samples with different labels. Therefore, LSH requires a large amount of hashing functions to produce a satisfactory performance, which is computationally intensive. 2) Because the number of hashing functions is large, LSH requires a significant amount of space for storing the hash table. 3) The performance of LSH is sensitive to a number of parameters; to obtain the optimal parameter, cross validation is usually applied which is time-consuming.

In this paper, to in order to alleviate the aforementioned problems of LSH, we propose a novel model for approximation nearest neighbors in high dimensions, termed Grassmann Hashing (GRASH). The procedure of GRASH is described as follows: 1) GRASH derives the Fisher faces [8] by applying Linear Discriminant Analysis (LDA) on original data; 2) GRASH selects the first d Fisher faces, i.e. vectors corresponding to first d largest eigenvalues; 3) from the derived d Fisher faces, GRASH traverses the combination of m Fisher faces (i.e. traverse C_d^m combinations in total) and construct the corresponding m dimension subspaces; GRASH also records the discriminant energy of every m dimension subspaces obtained; 4) GRASH applies the Grassmann distance metric together with discriminant energy to select the optimal k subspaces; these subspace are used for hashing; 5) GRASH applies non-uniform bucket size design to generate hashing codes; 6) GRASH applies c-approximation (i.e. c-bit distance) on the codes obtained in step 5) to approximates r-nearest neighbors in Hamming space.

The proposed GRASH enjoys the following merits compared against with LSH: 1) GRASH introduces the Grassmann metric to measure similarity between different subspaces, so the hashing function could produce have least overlap; 2) GRASH obtains the subspace candidates from LDA, so it could incorporate the discriminant information into the hashing functions; 3) GRASH extends LSH's 1-d hashing subspaces to m-d, i.e. it is a multidimensional extension of hashing approximation; 4) motivated by Lloyd-

Max quantization, GRASH applies non-uniform size bucket to generate hashing codes, so the distortion can be minimized.

The rest of this paper is organized as follows: in Section 2, we give a brief review on Grassmann manifold and Locality-Sensitive Hashing (LSH); in Section 3, we introduce our proposed Grassmann Hashing (GRASH) for approximating nearest neighbors in high dimensions, which consists of Section 3.1 hashing subspace candidates selection, Section 3.2 GRASH criterion for optimal subspace selection, Section 3.3 Hashing codes based on non-uniform bucket size design, Section 3.4 Hamming distance for GRASH; in Section 4, we show the experimental results comparing GRASH against other conventional hashing algorithms; in Section 5, we conclude the paper.

2. GRASSMANN METRIC AND LOCALITY SENSITIVE HASHING

In this section, we give a brief review on Grassmann metric and Locality Sensitive Hashing (LSH). In GRASH, we apply Grassmann metric to measure the distance between subspaces, and then we utilize the Grassmann metric and discriminant energy as criteria to select the optimal subspaces for hashing.

2.1. Grassmann Metric

Grassmann manifold $G(d,D)$ can be defined as the set of d-dimension linear subspace in R^D [2][3], i.e. every point on the Grassmann manifold refers to a subspace. Let us consider the space $\mathrm{R}_{\mathrm{D,d}}^{(0)}$ of all $D \times d$ matrices, i.e. $Y \in R^{D\times d}$. The group of transformation $\tilde{\mathrm{Y}} = \mathrm{YL}$,where L is a full rank $d \times d$ square matrix, defines a equivalence relation in $\mathrm{R}_{\mathrm{D,d}}^{(0)}$:

$$\begin{aligned} &Y_1 = Y_2 \ \ if \ \ \mathrm{span}(Y_1) = \mathrm{span}(Y_2) \\ &where \ Y_1, Y_2 \in R_{D,d}^{(0)} \end{aligned} \tag{1}$$

Therefore, the equivalence classes of $\mathrm{R}_{\mathrm{D,d}}^{(0)}$ are in one-to-one correspondence with the points on the Grassmann manifold $G(d,D)$, i.e. each point on the manifold is a subspace.

To measure the distance of two points on the Grassmann manifold is equivalent to measure the similarities between two subspaces. Principal angle [2][3] serves as a geometric measure between two subspaces. We consider two matrices orthogonormal $Y_1, Y_2 \in R^{D\times d}$ on the Grassmann manifold, the principal angles $0 \le \theta_1 \le \cdots \le \theta_d \le \pi/2$ between two subspaces $span(Y_1)$ and $span(Y_2)$ are recursively defined as:

$$\begin{aligned} &\cos\theta_k = \max_{u_k \in \mathrm{span}(Y_1)} \max_{v_k \in \mathrm{span}(Y_2)} u_k' v_k \\ &s.t. \ \ u_k' u_k = 1, \ v_k' v_k = 1, \\ &\quad u_k' u_i = 0, \ v_k' v_i = 0, \\ &\quad for \ i = 1, \ldots, k-1 \end{aligned} \tag{2}$$

where $(u_1, u_2, \ldots, u_d)$ and $(v_1, v_2, \ldots, v_d)$ are called principal vectors of the two subspaces, and θ_k is the kth smallest angle between two principal vectors u_k and v_k.

To compute the principal angles between θ two subspaces, we may apply Singular Value Decomposition (SVD) [2] on the product of the two matrices $Y_1'Y_2$, i.e.,

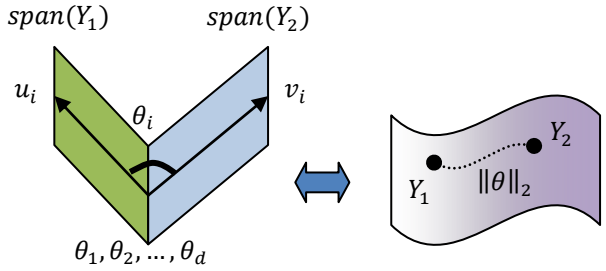
$$Y_1'Y_2 = USV' \tag{3}$$

where $U = [u_1, u_2, \ldots, u_d]$, $V = [v_1, v_2, \ldots, v_d]$ and $S = diag(\cos\theta_1, \ldots, \cos\theta_d)$. The cosine of principal angels θ , i.e. $\cos\theta_1, \ldots, \cos\theta_d$ are known as canonical correlations[17].

In this paper, to measure the distance between subspaces, we adopt the geodesic distance (or arc-length) [2][3], which is defined as follows:

$$d_{Arc}^2(Y_1, Y_2) = \sum_i \theta_i^2 \tag{4}$$

The geodesic distance is derived from the geometry of Grassmann manifold. It is the length of geodesic curve connecting two subspaces along the Grassmann surface. The geodesic distance is a distance measure, but it also satisfies the requirements of metric [2][3].



1) *principal angle in* R^D 2) *Grassmann distance in* $G(d,D)$

Figure 1 Principal angles θ in $\mathrm{R^D}$ and Grassman Distance in $\mathrm{G(d,D)}$. In the Grassmann manifold point of view, two subspaces $\mathrm{span(Y_1)}$ and $\mathrm{span(Y_2)}$ are two points on the manifold $\mathrm{G(d,D)}$, the geodesic distance between these two points on the manifold is $\mathrm{d(Y_1, Y_2)} = \|\theta\|_2$.

2.2. Locality Sensitive Hashing (LSH)

LSH is introduced as a probabilistic technique suitable for solving the approximate K-NN problem [1][5][6]. The basic insight of LSH is that it is possible to construct hash functions such that the probability of collision is much higher for samples that are close to each other than for those that are far apart. We consider the case of $\mathrm{R^D}$ with $\mathrm{L_2}$, LSH is defined as follows:

$$H(v) = \langle h_1(v), h_2(v), \ldots, h_M(v) \rangle \tag{5}$$

$$h_i(v) = \left\lfloor \frac{a_i \cdot v + b_i}{W} \right\rfloor, \quad i = 1, 2, \ldots, M \tag{6}$$

where $a_i \in R^D$ is a vector with entries selected at random from a Gaussian distribution, e.g. $N(0,1)$; b_i is drawn from the uniform distribution, e.g. $U[0, W)$; $\lfloor . \rfloor$ is the floor operation. The parameters M and W control the locality sensitivity of the hash function.

For the hashing functions to serve our purpose, we require that: 1) for points p and q in R^D that are close to each other:

$$P_H[h(p) = h(q)] \geq P_1 \quad \text{for } \|p - q\| \leq R_1 \tag{7}$$

where i.e. the probability the p and q fall into one bucket is "not so small"; 2) for points p and q in R^D that are far apart:

$$P_H[h(p) = h(q)] \leq P_2 \quad \text{for } \|p - q\| \geq R_2 = cR_1 \tag{8}$$

i.e. the probability the p and q fall into one bucket is small. In order for LSH family to be applicable, the following conditions have to be satisfied: 1) $c > 1$; 2) $P_1 > P_2$.

3. GRASSMANN HASHING FOR APPROXIMATING NEAREST NEIGHBOR SEARCHES

Because the hashing functions H(v) are randomly generated, LSH suffers from a number of drawbacks. First, LSH fails to incorporate the distance between hashing subspaces; moreover, LSH doesn't incorporate the discriminant information between samples. Therefore, the hashing code generated by LSH is inefficient; to produce a satisfactory performance, LSH requires a large amount of hashing functions, which is computation intensive. Second, because the number of hashing functions is large, LSH requires a significant amount of space for the hash table storage. Third, the performance of LSH is sensitive to a number of parameters; to obtain the optimal parameters, a time-consuming cross validation is usually applied.

Grassmann Hashing (GRASH) alleviates the aforementioned problems by incorporating the distance between projection subspaces and the discriminant information between samples from different classes. By introducing Grassmann metric and discriminant energy, GRASH measures the distance between different hashing subspaces and selects the optimal ones for hashing.

In this section, we introduce our proposed GRASH algorithm. The procedure of GRASH is illustrated in the following sub-sections.

3.1. Hashing Subspace Candidates

The first step of GRASH is to introduce a set of subspace candidates for hashing. In GRASH, we obtain the hashing subspace candidates from Linear Discriminant Analysis (LDA) [8], a well-known supervised learning algorithm. We apply Fisher faces as basic elements for constructing our subspace candidates. LDA obtains the optimal projection W as follows:

$$W = \arg\max_W \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|} = [w_1 \; w_2 \; \ldots \; w_n] \tag{9}$$

where S_B is the between-class scatter matrix, S_w is the within-class scatter matrix, W is the set of generalized eigenvectors of S_B and S_w [8]corresponding to the n descending-sorted eigenvalues $\{\lambda_i | i = 1,2, \ldots, n\}$, i.e.,

$$S_B w_i = \lambda_i S_W w_i, \quad i = 1,2,3,\ldots,n \tag{10}$$

Then we select the first d generalized eigenvectors, i.e. vectors corresponding to first d largest eigenvalues $W = [w_1, w_2, \ldots, w_d]$.

We now apply W to construct the hashing subspace candidates for GRASH. Let us consider the case of m-dimension hashing, i.e. each hashing function refers to a projection onto a R^m subspace. To obtain the R^m hashing subspaces, we traverse the combinations of the m Fisher faces out of n, i.e. in total C_d^m combinations; then we construct the corresponding hashing subspaces based on the m vectors, i.e. apply the m vectors as basis for the m-dimension hashing subspace [1][5][6]. The derived set of R^m hashing subspace is termed Hashing Subspace Candidate Set.

We record the discriminant energy of the derived Hashing Subspace Candidates, which is defined as follows:

$$E_t = \sum_i^m \lambda_i^2 \tag{11}$$

where t is the index of subspace, λ_i is the generalized eigenvalues corresponding to the Fisher face w_i derived from (9) and (10). By applying this energy definition, we associate every hashing subspace in the Hashing Subspace Candidates with a discriminant power. As we may discover, the subspace with largest discriminant energy is $E_{largest} = \lambda 12 + \lambda 22 + \ldots + \lambda m2$. This cost function is further used as part of criteria for selecting optimal hashing functions.

3.2. GRASH Criterion for Optimal Hashing Subspaces

To select the optimal m-dimension subspace for hashing, in GRASH we develop a criterion based on Grassmann metric and discriminant energy proposed in (11). To apply the criterion, initially we create a set termed Incoming Set U_{in} with one element $E_{largest}$, and we also duplicate the Hashing Subspace Candidate Set to create the Outcoming Set U_{out}. The criterion for selecting optimal hashing subspace is defined as follows:

$$i = \arg\max_i \left(\mu E_i + (1-\mu) \cdot \sum d_{Arc}^2(i, j)\right) \quad \forall j \in U_{in} \tag{12}$$

where i is the index of the selected subspace, μ is a tuning parameter between discriminant energy E and geodesic distance d_{Arc}, j is the index of subspaces in the Incoming

Set U_{in}. As we may observe, when $\mu = 1$, the GRASH criterion becomes discriminant power criterion; when $\mu = 0$, this criterion collapses to the Grassmann metric, i.e. geodesic distance in our case.

The selection process iterates as follows:

1. Initial $U_{in} = \{E_{largest}\}$, $U_{out} = U$,
2. for $p = 1 : k$,
3. if $U_{out} = \emptyset$ return;
4. else
5. find the optimal subspace i from U_{out} , based on (12);
6. add subspace i to U_{in};
7. subtract subspace i from U_{out}
8. end
9. end

where U is the universe set and k is the number of hashing functions defined by the users. By applying this criterion, we can incorporate the discriminant information regularized by the distance between different subspaces. As we may observe, in every iteration, we select the most promising subspaces from U_{out}, based on the discriminant information available. To speed up this iterative algorithm, we may build a tree data structure for storing E [13].

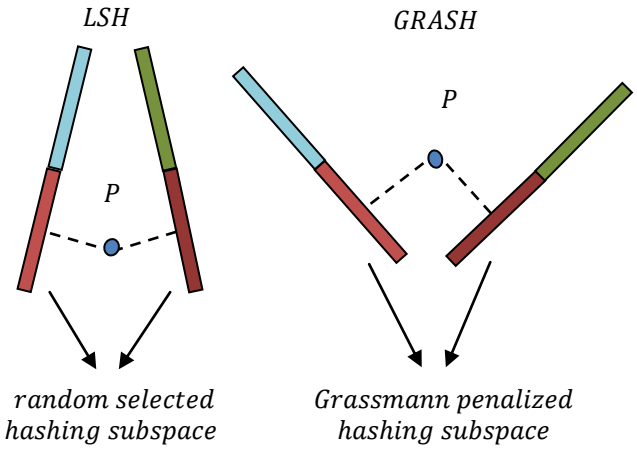


Figure 2 A comparison on hashing subspace between LSH and GRASH. LSH applies random hashing functions on the query; GRASH introduces Grassmann metric as penalization and selects the most "promising" subspaces. Projections onto these "promising" subspaces can give a more precise approximating in the original space.

3.3. Hashing Codes Based on Non-uniform Bucket Size Design

LSH applies uniform size buckets to generate hashing codes. This approach is most effective for data samples from uniform distribution, in which case the number of samples falling into each bucket is approximately equal. For most distributions, the number of samples falling into each bucket differs, so distortion rate is high [18].

To solve this problem, in GRASH, we apply the non-uniform size bucket to generate hashing codes. We apply Lloyd-Max [18] algorithm to define the quantizer threshold, which is achieved by minimizing the distortion error:

$$D = E\left[\left|x - \hat{x}\right|^2\right] \tag{13}$$

By applying Lloyd-Max algorithm, we implement non-uniform size bucket in a customized manner, i.e. each bucket is of a different size, but contains approximately equal number of samples. The coding process is illustrated as follows: first, we project samples onto the subspaces selected in Section 3.1 and Section 3.2; second, we define r, i.e. the number of bits for each hashing vector; third, sort the projected samples along the hashing vector, and cluster them into 2^r regions.

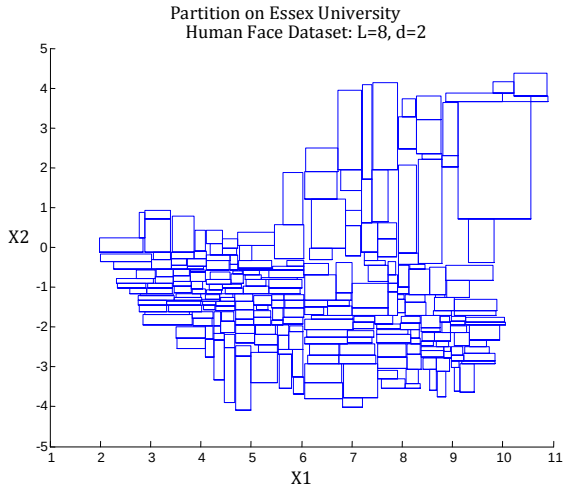


Figure 3 Coding by non-uniform bucket size partition on Essex University Human Face dataset [16][11][13]. Each blue box represents a hash bucket projected onto the 2D subspace.

Figure 3 shows the partition result of non-uniform bucket size on Essex University Human Face dataset. The partition is conducted in the following manner: 1) apply the selected projections on the whole data set, obtain the set of basis $A = [a_1, a_2, \ldots a_d]$; 2) project all sample points on the maximum variance basis a_1, find the median value of the projected samples m_1, and split the whole collection of data along a_1 at m_1, i.e. split the current node into left and right children; 3) starting from $i = 2$, for each left and right child, project the whole collection of data along the i-th maximum variance basis a_i, find the median value m_i, and split all the children at m_i, generate code correspondingly; 4) increment i, repeat 3) until some predefined criteria for number of levels, or the number of samples in the leaf node is satisfied; 5) obtain codes for the whole dataset. At each node, a bucket represented is computed and recorded.

The non-uniform size bucket design for GRASH enjoys the following merits: 1) it is a customized query-driven design, the number of samples falling into each region is approximately equal; therefore, the distortion rate can be minimized and the hashing efficiency can be improved; 2) by introducing the non-uniform bucket size, we can incorporate more amount of discriminant information by increasing the "degree of freedom" on coding.

3.4. Hamming Distance for GRASH

Similar to LSH, in GRASH, the hashing functions is given :

$$g(p) = < h_1(p), h_2(p), ..., h_k(p) > \quad (14)$$

where k is the number of hashing subspaces. We define the probability of collision in Hamming space as follows:

$$P_H[h(p) = h(q)] = 1 - D(p,q)/mkr \quad (15)$$

where $D(p,q)$ is the distance between p and q in Hamming space, m is the dimension of hashing subspaces. As we may observe, mkr refers to the length of the hashed codes.

4. EXPERIMENTS

To confirm the validity of our proposed GRASH, we apply this algorithm on a number of datasets and compare our approximation performance against other models. We performed experiments on two datasets. The first dataset is the human face datasets, which contains 6,680 faces of 417 individuals; each image is represented as a point in 400D space. The second dataset is the MSRA-MM dataset, which contains 65,443 images of 68 classes; each image is represented by its 899D feature.

Implementation We implement our proposed algorithm as specified in section 3. We first apply LDA on the two datasets, and we then obtain the m-dimension subspace with largest discrimination energy, and last we apply the subspace selection criterion developed in section 3.2 to derive the optimal hashing subspaces.

Performance Measure In this paper, we apply the intersection rate to measure the performance of GRASH against that of other algorithms. We compute number of nearest neighbors obtained by GRASH and compare them against the ground-truth nearest neighbors in the original Euclidean space. In this paper, we define the precision as follows:

$$I = \frac{1}{|Q|}\sum_{q\in Q}\frac{\left\|U_{q,GRASH} \cap U_{q,*}\right\|}{\left\|U_{q,*}\right\|} \quad (16)$$

where $U_{q,GRASH}$ is the set of nearest neighbors returned by GRASH, and $U_{q,*}$ is the set returned from the ground truth.

4.1. Human Face Dataset

In this section, we apply GRASH to approximate nearest neighbors on biometric dataset. To test our proposed model with a large number of training samples, we assemble a large human face dataset by combining several well-known human face dataset, i.e. YALE [8], ORL[14], FERET[17], UMIST[15], and Essex[16]. We select 165 faces from YALE, 400 faces from ORL, 700 faces from FERET, 575 faces from UMIST and 4,840 faces from Essex. In total, we have 6,880 faces for 417 individuals. All images are normalized to 20×20 pixel arrays with 256 gray levels per pixel. The 400D pixel is treated as the feature information for all samples.

On this dataset, we apply intersection rate defined in (16) to measure the performance of GRASH again LSH. For every hashing subspace, we apply 1-bit, 2-bit and 4-bit (per hashing function) coding scheme and compare the intersection rate (i.e. the rate defined in (16)) against that of LSH.

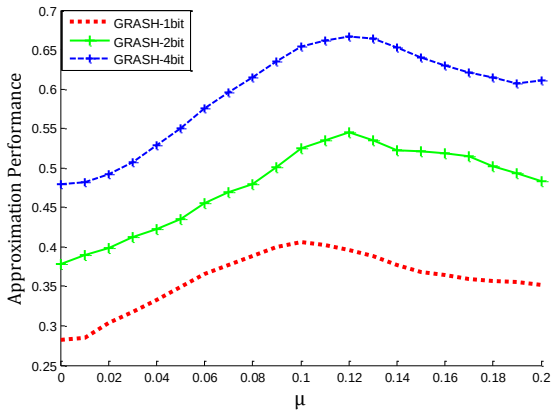


Figure 4 The approximation performance (i.e. the intersection rate) vs μ. In this paper, we apply cross validation to record the optimal μ. The recorded rates are averaged on 25 trials; the number of hashing functions is 20. Approximation based on 8-NNS is conducted.

In Table 1, we adopt 20 hashing functions and conduct 1-bit, 2-bit and 4-bit coding on each hashing function, i.e. in total 20 bits, 40 bits and 80 bits coding for each sample. Figure 4 shows the approximation performance versus μ on the Human Face Dataset. Empirically, the optimal μ lies in the domain of [0.05, 0.2].

4.2. MSRA-MM Dataset

MSRA-MM dataset[9] consists of two sub-datasets: an image dataset and a video dataset. In this paper, we utilize the image dataset to test the performance of GRASH. The image dataset contains 68 classes, each of which consists of around 1,000 images, and in total 65,443 images. All the images are collected from the query log of Microsoft Live Search. For this image dataset, the original images not provided due the copyright issue. However, a set of features are available including: 1) 225D block-wise color moment; 2) 64D HSV color histogram; 3) 256D RGB color histogram; 4) 144D color correlogram; 5) 75D edge distribution histogram; 6) 128D wavelet texture; 7) 7D face features. Therefore, for each sample, we have 899D data in total. In

	4-NNS	8-NNS	16-NNS	32-NNS
LSH-1bit	23.9%	28.8%	33.6%	35.1%
LSH-2bit	31.5%	34.6%	39.3%	39.8%
LSH-4bit	40.6%	45.7%	51.2%	55.1%
GRASH-1bit	39.3%	42.4%	49.7%	53.2%
GRASH-2bit	52.8%	55.8%	68.3%	72.3%
GRASH-4bit	**63.9%**	**69.7%**	**73.6%**	**80.3%**

Table 1 Intersection rate of LSH against that of GRASH. The rate is averaged on 25 trials; for each trial, training samples are selected randomly and the average rate is recorded. μ is obtained by cross validation. The No. of hashing functions is 20.

this paper, we select 10 classes, i.e., *background, bird, dragon, email, fruit, hairstyle, panda, people, tree, youtube.*

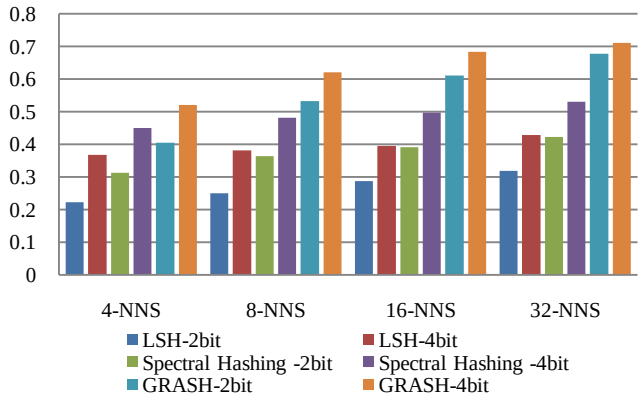


Figure 5 Approximation performance of GRASH against LSH and Spectral Hashing [7].The recorded rates are averaged on 25 trials; the number of hashing functions is 20.

Figure 5 shows the approximation performance (i.e. intersection rate) of GRASH against that of LSH and Spectral Hashing. Similar to Table 1, we adopt performance measure in (16) and apply 2-bit/4-bit hashing based on LSH, Spectral Hashing and GRASH.

5. CONCLUSIONS

Locality-Sensitive Hashing (LSH) approximates nearest neighbors by projecting original data into low-dimensional subspaces. However, because LSH applies random projections for hashing, it fails to incorporate discriminant information between different hashing subspaces. Therefore LSH requires a large amount of hashing functions to generate satisfactory performance. To alleviate this problem, we propose Grassmann Hashing (GRASH) for approximating nearest neighbors. GRASH applies Grassmann metric and discriminant energy as criteria for selecting hashing subspaces, so it could significantly reduce the number of hashing functions required to produce a satisfactory approximation result. Experimental results confirm the validity of our proposed algorithms.

In the future, we plan to introduce our proposed algorithm to other conventional subspace learning methods e.g. LPP[12] for approximating nearest neighbors in high dimensions.

6. REFERENCE

[1] A. Gionis, P. Indyk, R. Motwani. Similarity Search in High Dimensions via Hashing. VLDB 1999.

[2] J. Hamm, D. D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. ICML 2008

[3] J. Hamm. Subspace-Based Learning with Grassmann Kernels, PhD Thesis, 2008.

[4] W. Dong, Z. Wang, W. Josephson, M. Charikar, K. Li. Modeling LSH for Performance Tuning. ICIKM 2008

[5] M. Slaney, M. Casey. Locailtiy-Sensitive Hashing for Finding Nearest neighbors. IEEE SP-M 2008

[6] A. Andoni, P. Indyk. Near-Optimal Hashing for Algorithms for Approximate Nearest Neighbor in High Dimensions. ASFCS 2006.

[7] Y. Weiss, A. Torralba, R. Fergus. Spectral Hashing. NIPS 2005

[8] P. Belhumeur, J. Hespanha, D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection, IEEE T-PAMI 1997

[9] H. Li, M. Wang, X. Hua, MSRA-MM 2.0: A Large-Scale Web Multimedia Dataset, ICDM-Workshop 2009

[10]Y. Ke, R. Sukthankar, L. Huston. Efficient Near-duplicate Detection and Sub-image Retrieval. ACM Multimedia 2004

[11] Y. Fu, Z. Li, J. Yuan, Y. Wu, and Thomas S. Huang, "Locality vs. Globality: Query-Driven Localized Linear Models for Facial Image Computing," IEEE T-CSVT, 2008.

[12] X. He, P. Niyogi. Locality Preserving Projections. NIPS 2003

[13] Z. Li, L. Gao, A. K. Katsaggelos. Locally Embedded Linear Subspaces for Efficient Video Indexing and Retrieval. ICME 2006

[14] F. Samaria, A.C. Pentaland. Parameterization of a stochastic model for human face identification IEEE Workshop on Vision and Applications 1994

[15] D. B. Graham, N. M. Allinson. Characterizing virtual eigensignatures for general purpose face recognition. Face Recognition: From Theory to Application. Computer & Systems Sciences, 1998.

[16] D.Hond, L.Spacek. Distinctive Descriptions for Face Processing. BMVC, 1997.

[17] P. J. Philiips, H. Moon, P. J. Rauss, and S. Rizvi. "The FERET evaluation methodology for face recognition algorithms. IEEE Trans. PAMI, 2000.

[18] M. R. Garey, D.S. Johnsom, H.S. Witsenhausen. Complexity of the Generalized Lloyd-Max Problem. IEEE-TIT 1982