Fast Subspace Search via Grassmannian Based Hashing

Xu Wang
Math Department, University of Minnesota
wang1591@umn.edu

Stefan Atev
Proto Labs, Inc.
stefan.atev@gmail.com

John Wright
EE Department, Columbia University
johnwright@ee.columbia.edu

Gilad Lerman
Math Department, University of Minnesota
lerman@umn.edu

Abstract

The problem of efficiently deciding which of a database of models is most similar to a given input query arises throughout modern computer vision. Motivated by applications in recognition, image retrieval and optimization, there has been significant recent interest in the variant of this problem in which the database models are linear subspaces and the input is either a point or a subspace. Current approaches to this problem have poor scaling in high dimensions, and may not guarantee sublinear query complexity. We present a new approach to approximate nearest subspace search, based on a simple, new locality sensitive hash for subspaces. Our approach allows point-to-subspace query for a database of subspaces of arbitrary dimension d, in a time that depends sublinearly on the number of subspaces in the database. The query complexity of our algorithm is linear in the ambient dimension D, allowing it to be directly applied to high-dimensional imagery data. Numerical experiments on model problems in image repatching and automatic face recognition confirm the advantages of our algorithm in terms of both speed and accuracy.

1. Introduction

Given a very large database of models, how can we efficiently determine which one that best fits a given input query? This basic question arises repeatedly in computer vision applications such as visual recognition, categorization, image retrieval and beyond. These applications pose two general challenges to the algorithm designer: imagery data (and their features) are typically *high-dimensional*, and databases arising in applications can be very *large scale*.

The large scale often precludes simply comparing the query to each of the models in any reasonable amount of time. Instead, researchers typically resort to more sophisticated *approximate nearest neighbor* techniques, whose query time is sublinear in the size of the database. For the case in which the query is a vector and the database is also a collection of vectors, these techniques are very well-developed, in both theory and practice [5, 6, 11, 21].

However, data in computer vision problems often have rich physical or geometric structure, which may not be well-encoded using point models. For example, photometric or textural properties of a collection of images can often be better represented using linear or affine *subspaces*, rather than a simple point model. In the *approximate nearest subspace* problem, we are given a collection of linear subspaces. The goal is to efficiently determine which of the database subspaces is closest to the input [3]. Good solutions to this problem would allow us to efficiently query large databases which contain much richer representations.

In contrast to approximate nearest neighbor, both the theory and practice of approximate nearest subspace are still developing. The most general known approach is due to Basri et. al. [3]. It maps each subspace $S \subseteq \mathbb{R}^D$ to its orthogonal projection matrix $\mathbf{P}_S$, and then applies an approximate nearest neighbor algorithm to the projection matrices. The advantage of this approach is that it cleanly reduces the subspace problem to the better-understood point search problem. However, because the projection matrix has size $\Theta(D^2)$, the algorithm's performance suffers in high dimensions.[1] Moreover, the mapping from a subspace to an orthoprojector does not preserve distances (for subspaces of different dimensions), and so performance guarantees for the approximate nearest neighbor algorithm may not pull back to the approximate nearest subspace problem. Algorithms with sublinear query time, but exponential dependence on dimension have also been introduced in [16].

Motivated in part by [3], there has been a flurry of recent work on special cases of this problem. For example the

[1] [3] suggest using random projections after lifting as one means of controlling the complexity.

case in which the query is a point and the database contains hyperplanes (of dimension $d = D - 1$) has been studied in connection to active learning and large-scale regression [13]. Various approaches based on locality sensitive hashing have been proposed [13, 15, 18, 19]. While various technical obstacles prevent these approaches from guaranteeing sublinear query complexity over all inputs, they have been used effectively in various practical vision problems. In the algorithms community, there is also dedicated work on the special case in which the queries are points and the database consists of affine lines ($d = 1$). For example, Andoni et. al. produce a data structure for this problem that has query time $O(D^3 n^{1/2+t})$ and space complexity $D^2 n^{O(1/(c-1)^2+1/t^2)}$, for any $t > 0$ [2]. Again, the query time $O(D^3)$ could be problematic in large dimensions.

Moreover, many of the most interesting models for computer vision have a dimension d that falls somewhere in between 1 and $D - 1$. For example, linear subspaces spanned by images taken under varying lighting may have dimension between 3 and 9, depending on the properties of the object [4]. Local image patches also typically lie near subspaces of dimension higher than one [22, 23]. So, despite the above progress, there is still a need for algorithms that can guarantee a query time that is sublinear in the number of models n, have good (linear or sublinear) dependence on the ambient dimension D, and can handle the case when the input is a point and the database consists of subspaces of arbitrary dimension d.

Contributions. In this paper, we provide a solution of the approximate nearest subspace (ANS) search problem based on the notion of *locality sensitive hashing* (see e.g., [11]). We consider only linear subspaces. Our theoretical guarantees for the sub-linear complexity and preprocessing space of our solution distinguish between three types of searches: line-line query (this is equivalent to point-line query as explained in §2; it is also equivalent with line-point and point-point queries when the points lie on the sphere); line-subspace query (this is equivalent to point-subspace query as explained in §2); and subspace-subspace query (for subspaces of the same dimension). For all of these searches, our preprocessing space is $O(n^{1+\rho} + nDd)$ and query time is $O(Ddn^{\rho})$, where d is the largest dimension of subspaces among both query elements and the database elements, D is the ambient dimension and $\rho < 1$.

Nevertheless, the precise formulations and their corresponding estimates are different for the three types of searches. For the subspace-subspace query (with dimension of subspaces greater than 1), the above estimate for the query time holds as long as for each query element there is a sufficiently close element in the database and the approximation constant is sufficiently large (with respect to the subspace dimension). For line-line search we can obtain better estimates of the parameters, in particular, asymptotic estimate of ρ (for a special setting).

Our theoretical setting is designed to address recognition problems. For example, our unorthodox restriction on the maximal distance between query element and the database (this appears only in some of our statements) can often be met in practice, where query points may be contained in or be sufficiently close to the database. We confirmed in practice the competitive speed and accuracy of our proposed solution on model problems in image repatching and automatic face recognition.

Organization of this paper. In §2 we introduce notational conventions and adapt the notion of locality sensitive hashing to the ANS problem. We then generalize a well-known theoretical framework claiming that a locality sensitive hashing family gives rise to a search algorithm with sub-linear time. In §3, we propose a concrete hashing family for the Grassmanian manifold $\mathrm{G}(D, d)$ and for the union $\mathrm{G}(D, 1) \cup \mathrm{G}(D, d)$. We then formulate the main theorems of this work detailing the quality of the basic sub-linear search procedure in each one of the three types of searches described above. The details of the ANS algorithm resulting from the locality sensitive hashing family we proposed are outlines in §4, whereas §5 compares our ANS algorithm method with the ANS algorithm of Basri et al. [3] on model problems in image repatching and automatic face recognition.

2. Problem Formulation and Preliminaries

The Grassmannian. Let $\mathrm{G}(D, d)$ denote the *Grassmanian manifold*, i.e., the space of all d-dimensional linear subspaces of $\mathbb{R}^D$. If $0 < d_1 \leq d_2 < D$, $L_1 \in \mathrm{G}(D, d_1)$ and $L_2 \in \mathrm{G}(D, d_2)$, the *principal angles* $\theta_1 \geq ... \geq \theta_{d_1}$ between L_1 and L_2 can be defined as follows [9]: Let $\mathbf{Q}_{L_1}$ and $\mathbf{Q}_{L_2}$ be matrices whose columns are orthonormal bases for L_1 and L_2, respectively. For $i = 1, \ldots, d_1$ let $\sigma_i(\boldsymbol{Q}_{L_1}^T \boldsymbol{Q}_{L_2})$ denote the i-th largest singular value of the matrix $\boldsymbol{Q}_{L_1}^T \boldsymbol{Q}_{L_2}$. The principal angles $\pi/2 \geq \theta_1 \geq \theta_2 \geq \cdots \geq \theta_d \geq 0$, are[2]

$$\theta_i = \arccos(\sigma_{d-i}(\boldsymbol{Q}_{L_1}^T \boldsymbol{Q}_{L_2})), \quad i = 1, \ldots, d_1. \tag{1}$$

Using these angles, the "distance" between L_1 and L_2 is

$$\mathrm{dist}_{\mathrm{G}}(L_1, L_2) = \left(\sum_{i=1}^{d_1} \theta_i^2 \right)^{1/2}. \tag{2}$$

If $d_1 = d_2 = d$, it is a metric; where if $d_1 \neq d_2$, it is still a good measure of proximity. For example, if $d_1 = 1$, then $\mathrm{dist}_{\mathrm{G}}(L_1, L_2)$ is the elevation angle between the line L_1 and the subspace L_2.

[2] Here, we order the principal angles decreasingly, unlike the common arrangement [9] (§12.4.3).

Approximate Subspace Search. Motivated by the approximate nearest point search in [1], we define the approximate nearest subspace search problem as follows:

Definition 2.1. (R,c)**-approximate subspace search:** *Let X be a set of d_2-dimensional subspaces in $\mathbb{R}^D$ and R, c, δ be positive numbers. A search algorithm is called* (R,c)-approximate subspace search *if it fulfills the following requirement. Given a query subspace L of dimension d_1, if there is an element L' in X s.t.* $\mathrm{dist}_{\mathrm{G}}(L,L') \leq \mathrm{R}$*, then, an element L'' in X with* $\mathrm{dist}_{\mathrm{G}}(L'',L) < cR$ *is returned with probability* $1-\delta$.

For several applications, the most interesting query problem is the point-subspace query, the query is a point in $\mathbb{R}^D$ and the database is a subset of $\mathrm{G}(D,d)$. By connecting points with the origin to obtain lines, the point-subspace query problem is reduced to the (R,c)-approximate subspace search problem with $d_1 = 1$ (where we denote d_2 by d). However, instead of measuring the Euclidean distance of the query point to the subspace, we measure the equivalent "distance", $\mathrm{dist}_{\mathrm{G}}$, between the line through the query point and the subspace.

The use of this equivalent "distance" results in a point-subspace query. Indeed, assume that the query point $\boldsymbol{x}_0$ has a principal angle θ_0 and Euclidean distance $||\boldsymbol{x}_0||_2 \sin\theta_0$ w.r.t. the nearest subspace. Our algorithm returns a subspace which has principal angle $c\theta_0$ and Euclidean distance $||\boldsymbol{x}_0||_2 \sin(c\theta_0) < c||\boldsymbol{x}_0||_2 \sin\theta_0$ with the query (the inequality is true for any $c > 1$). This means the solution of the line-subspace query problem is also a solution for the corresponding approximate point-subspace query problem.

Locality Sensitive Hashing. Following [11], we apply the notion of locality sensitive hashing (LSH) family to the subspace search situation. We generalize the definition of [11] for LSH as follows:

Definition 2.2. Locality sensitive hashing family for (*X*,*Q*,*F*): *Let X be a database, Q be a query set and F be a mapping from $X \times Q$ to $[0,\infty)$, which aims to measure the nearness between query and database points. A family $\mathcal{H}$ of functions on $X \cup Q$ with a probability measure $\mathbb{P}$ is called (R, cR, p_1, p_2)-sensitive for (X,Q,F) if for any $L_1 \in X, L_2 \in Q$:*

$$\begin{aligned} &\mathbb{P}[h \in \mathcal{H} | h(L_1) = h(L_2)] \geq p_1, \text{ if } F(L_1,L_2) \leq R; \\ &\mathbb{P}[h \in \mathcal{H} | h(L_1) = h(L_2)] \leq p_2, \text{ if } F(L_1,L_2) \geq cR. \end{aligned} \tag{3}$$

We require that $p_1 > p_2$ in order for the corresponding algorithm to work.

We are interested in two cases. The first case is when $X, Q \subset \mathrm{G}(D,d)$ and $F = \mathrm{dist}_{\mathrm{G}}$. This corresponds to the approximate subspace-subspace query problem. In this case, the definition of LSH family in [11] coincides with Definition 2.2. The second case is when $X \subset \mathrm{G}(D,d)$, $Q \subset \mathrm{G}(D,1)$ and $F = \mathrm{dist}_{\mathrm{G}}$. This corresponds to the approximate point-subspace (equivalently line-subspace) search problem.

The following theorem states that using the general LSH family of Definition 2.2, we can easily construct a corresponding locality hashing algorithm. Thus our main issue is to form an LSH family. This theorem is an immediate generalization of a theorem in [11, page 17]. Its proof is the same while replacing the neighborhood $B(q,r)$ of a query q with the set $\{x \in X | F(x,q) < r\}$.

Theorem 2.3. *Let X be a database, Q be a query set, F a mapping from $X \times Q$ to $[0,\infty)$ and denote by n the size of X. If there is a (R, cR, p_1, p_2)-sensitive family $\mathcal{H}$ for (X,Q,F), where $p_1, p_2 \in (0,1)$, then one can randomly draw from $\mathcal{H}$ to form a set $\mathcal{G}$ of vector-valued hash functions from X to $\{0,1\}^{\lceil \log_{1/p_2} n \rceil}$ such that for $\rho = \log(p_1)/\log(p_2)$:*

- *For any query point in Q, the corresponding basic hashing procedure with $\mathcal{G}$ requires at most $O(n^\rho/p_1)$ evaluations of the hash functions from $\mathcal{G}$.*
- *The number of elements in $\mathcal{G}$ is at most $O(n^\rho/p_1)$. Thus evaluating at n points requires storage of order $O(n^{1+\rho}/p_1)$. The total storage is the sum of this storage and the storage of the original data.*

The failure probability δ of the data structure is at most $1/3 + 1/e$ (e is Euler's number).

We remark that any LSH algorithm may return an empty set, unlike tree-based algorithms (see e.g., [3]).

3. Hashing Linear Subspaces

In this section, we describe a general hashing scheme that applies to approximate nearest subspace search problems in which the database consists of d_2-dimensional subspaces, and the query is a d_1-dimensional subspace. We claim (and prove in the supplementary material) that this scheme gives a locality sensitive hashing family for two cases of practical importance: $d_1 = d_2$ (subspace-subspace query) and $d_1 = 1, d_2 > 1$ (line-subspace query).

We generate the hashing scheme simply by thresholding the angle between the subspace L and a randomly generated line $\ell \in \mathrm{G}(D,1)$:

Definition 3.1. *Let $Q = \mathrm{G}(D,d_1)$ and $X = \mathrm{G}(D,d_2)$. For each line $\ell \in \mathrm{G}(D,1)$ and $0 < \theta_0 < \pi/6$, we associate a function $h_{\ell,\theta_0} : X \cup Q \longrightarrow \{0,1\}$, via*

$$h_{\ell,\theta_0}(L) = \begin{cases} 0, & \mathrm{dist}_{\mathrm{G}}(\ell, L) > \theta_0, \\ 1, & \mathrm{dist}_{\mathrm{G}}(\ell, L) \leq \theta_0. \end{cases} \tag{4}$$

Let $\mathcal{H}_{\theta_0}(d_1, d_2, D)$ denote the set of such functions h_{ℓ,θ_0}, with the uniform measure on $\mathrm{G}(D,1)$. Also denote $\mathcal{H}_{\theta_0}(d, D) = \mathcal{H}_{\theta_0}(d, d, D)$.

Main Properties. In practical applications such as recognition, it is valuable to allow the database to consist of subspaces (say, one subspace per subject). Our construction also yields sublinear-time algorithms for the important case of point-subspace (or equivalently line-subspace) query and consequently for the line-line search:

Theorem 3.2. *For any D, $d \leq D$, $c > 1$, $R > 0$ and $0 < \theta_0 < \pi/6$, there exist fixed positive real numbers $0 < p_2 < p_1 < 1$, such that $\mathcal{H}_{\theta_0}(1, d, D)$ is a (R, cR, p_1, p_2) locality sensitive family on $(\mathrm{G}(D,d), \mathrm{G}(D,1), \mathrm{dist}_{\mathrm{G}})$.*

Finally, our construction extends to query subspaces of higher dimensions, i.e., $Q = X = \mathrm{G}(D,d)$, with one caveat: We require R to be small ($R < R_0(c,\theta_0) \ll 1$), and c to be large ($c > \sqrt{d}$):

Theorem 3.3. *For any fixed $0 < \theta_0 < \pi/6$ and $c > \sqrt{d}$, there exists $R_0(c,\theta_0) \ll 1$ such that for any $R < R_0$, there are positive real numbers $0 < p_2 < p_1 < 1$ depending on c and R, such that $\mathcal{H}_{\theta_0}(d, D)$ with the induced uniform measure on it is $(R, cR, p_1, p_2)-$locality sensitive hashing family over $(\mathrm{G}(D,d), \mathrm{G}(D,d), \mathrm{dist}_{\mathrm{G}})$.*

Algorithmic Implications. The sub-linear time in our theoretical guarantees depends on the exponent $\rho = \log(p_1)/\log(p_2)$. In general, ρ depends on the parameters D, d, c, R and θ_0. An integral expression for the exponent ρ is provided in the supplementary material. In practice, ρ can be estimated by numerically integrating this expression and it can be numerically optimized by noticing the effect of various values of c, R and θ_0 on its expression (see supplemental material). The choice of R depends on the distribution of the query points within the database and the estimate of ρ improves as R decreases. Ideally, each query element needs to be within distance R to the database. In many practical cases, the query points are contained or sufficiently close to the database and R can be sufficiently small. The "precision" parameter c is chosen according to practical needs (in the case where the query elements are contained in the database it can be arbitrarily large). To make p_2 as small as possible, θ_0 should be chosen to be $\pi/6$ (that is, maximal) and empirical experiments support this choice.

In two situations, we can assert the asymptotic behavior of the exponent ρ. The first case is when both Q and X are subsets of $\mathrm{G}(D,1)$ (if the points are on the sphere, then it translates to point-point query, that is, nearest neighbor search), D approaches infinity and the query elements are sufficiently close to the database.

Theorem 3.4. *If $Q = X = \mathrm{G}(D,1)$, $R = \alpha/\sqrt{D}$ ($\alpha > 0$), $cR = O(1)$ and $0 < \theta_0 < \pi/6$ is fixed, then*

$$\lim_{D\to\infty} \rho(D, d, c, R, \theta_0) \leq 1/(1 + e^{\alpha^2/2}). \tag{5}$$

The second case is when $Q \subset X \subset \mathrm{G}(D,d)$ and n approaches infinity. Here ρ can be arbitrarily small as follows.

Theorem 3.5. *Assume that $Q \subset X \subset \mathrm{G}(D,d)$. For any $\rho > 0$, there is a locality sensitive hashing scheme to retrieve points from X, whose query time is at most $O(Ddn^{\rho})$.*

Theorem 3.5 follows from two observations. The first one is that since every query is in the database, we can pick R to be very small and c to be large while keeping cR small. The second observation is that since c is large, the ratio between the logarithms of p_1 and p_2 can be made sufficiently small.

4. Algorithm: Grassmanian-based Locality Hashing (GLH)

We propose the Grassmanian-based locality hashing (GLH) algorithm, which exploits standard techniques from locality sensitive hashing to convert the hashing scheme described in the previous section into an efficient near-subspace search. In offline preprocessing, we generate a hash table and assign database points to it. This process is described in Algorithm 1, where $\mathbb{S}^{D-1}$ denotes the $(D-1)$-dimensional unit sphere.

Algorithm 1 Preprocessing

Input: $S, K \in \mathbb{N}$, $0 < \theta_0 \leq \pi/6$ ($\theta_0 \in \mathbb{R}$) and a database $X \subset \mathrm{G}(D,d)$.

Output: Keys $\{\mathrm{Key}_{j,k}^{L}\}_{1\leq j\leq S, 1\leq k\leq K}^{L\in X}$ ($Key_{j,k}^{L} \in \mathbb{R}$) and random vectors $\{\boldsymbol{x}_{j,k}\}_{1\leq j\leq S,1\leq k\leq K}$ ($\boldsymbol{x}_{j,k} \in \mathbb{S}^{D-1}$).

Steps:

for $1 \leq j \leq S$ **do**
  for $1 \leq k \leq K$ **do**
    • Randomly choose $\boldsymbol{x}_{j,k}$ from $\mathbb{S}^{D-1}$ according to the uniform measure
    • $l_{j,k} := Span(\boldsymbol{x}_{j,k})$
  end for
  for $L \in X$ **do**
    • $\mathrm{Key}_{j,k}^{L} = h_{l_{j,k},\theta_0}(L)$, $k = 1, \ldots, K$
  end for
end for
return $\{\mathrm{Key}_{j,k}^{L}\}_{1\leq j\leq S,1\leq k\leq K}^{L\in X}$ and $\{\boldsymbol{x}_{j,k}\}_{1\leq j\leq S,1\leq k\leq K}$

At query time, we are given a new input subspace L and an input parameter $N \in \mathbb{N}$ (together with the given database and as well as the hash table and random vectors, which were generated in the preprocessing step). We consider as possible candidates at most N subspaces in the database,

which hash to the same bin as L, and perform an exhaustive search within this set. This procedure is described below in Algorithm 2.

Algorithm 2 Locality sensitive hashing for subspace search

Input: A query subspace $L \in \mathrm{G}(D,1) \cup \mathrm{G}(D,d)$, a database $X \subset \mathrm{G}(D,d)$, $S, K, N \in \mathbb{N}$, random vectors $\{\boldsymbol{x}_{j,k}\}_{1\le j\le S, 1\le k\le K}$ and keys $\{\mathrm{Key}_{j,k}^{L}\}_{1\le j\le S,1\le k\le K}^{L\in X}$.

Output: An (R,c)-approximate nearest subspace of L.

Steps:

- $l_{j,k} := Span(\boldsymbol{x}_{j,k})$, $1 \le j \le S$, $1 \le k \le K$
- $\mathcal{L} = \varnothing$
- Count=0

for $1 \le j \le S$ **do**

  if Count $\le N$ **then**

    - $\mathrm{Key}^{L} = (h_{l_{j,1},\theta_0}(L), ..., h_{l_{j,K},\theta_0}(L))$
    - Search for $L' \in X$, s.t. $(\mathrm{Key}_{j,1}^{L'}, ..., \mathrm{Key}_{j,K}^{L'}) = \mathrm{Key}^{L}$
    - If such L' exists and $\mathrm{dist}_{\mathrm{G}}(L', L) \le cR$, then Count=Count+1 and $\mathcal{L} = \mathcal{L} \cup \{L'\}$

  end if

end for

return The subspace in $\mathcal{L}$ closest to L (if exists)

5. Experiments

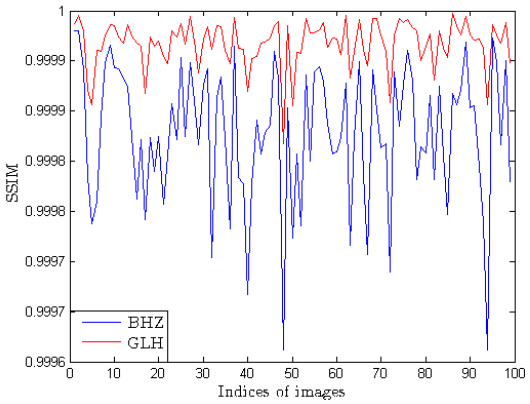
The scheme of Basri, Hassner and Zelnik-Manor (BHZ) [3] is the most general known approach for the approximate subspace search problem. To evaluate the performance of our GLH scheme, we carry out experiments on two model problems and compare the results with results by BHZ. In the first problem, patch-based image reconstruction, the ambient dimension is relatively low. Both schemes perform much faster than exact search. While the speed of the GLH scheme and the BHZ scheme are comparable, the performance of GLH is more stable across different images. Moreover, GLH has higher accuracy over BHZ for all images. The data sets of the second problem contain cropped images of faces under different illuminating conditions, where subspaces are formed by spans of vectorized images of the same face. In this setting the ambient dimension is relatively high and the GLH scheme obtains reliable results, while the BHZ scheme fails most of the time.

5.1. Image Approximation

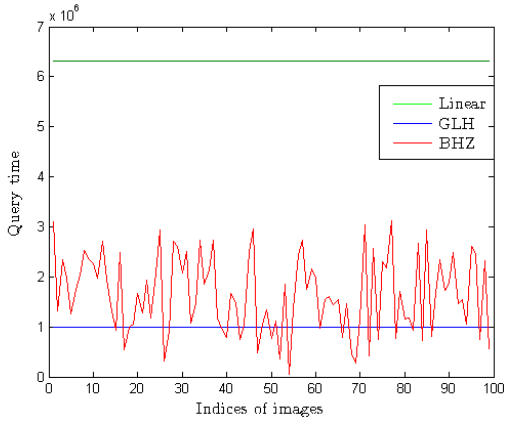
We follow Basri et al. [3] (with few technical modifications) and try to reconstruct images using a dictionary of subspaces constructed from an arbitrarily chosen image. We use the Berkeley segmentation database [17], which contains 100 test images of size 481×321.

We randomly pick one image from this database and also randomly select 1000 pixels from it. Then, 16 different, overlapping 5×5 patches around each pixel are used to produce a $k = 4$ dimensional subspace by taking principal components. This produces a database of 1,000 subspaces. Each of the 100 images is subdivided into nonoverlapping 5×5 patches. For each patch, we search for the closest subspace in the database by using both the GLH scheme and the BHZ scheme. We take the projection of the patch onto the selected subspace as its approximation.

To measure the quality of reconstruction, we use the structural similarity (SSIM) index [20], which effectively detects the distortion of an image from another image. We recall that $-1 \le \mathrm{SSIM} \le 1$ and $\mathrm{SSIM} = 1$ if and only if the two images are identical. Figure 1(a) shows that the



(a) SSIM indices



(b) Running time

Figure 1. Structural similarities (between original and repatched images) and number of evaluations: 100 test images from the Berkeley Segmentation Database are used in this experiment. Figure(a) shows the SSIM indices, which reflect similarity between the original images and their repatched images; the SSIM indices for GLH and BHZ are in red (upper curve) and blue (lower curve) respectively. Figure (b) shows the number of evaluation needed to repatch each image, where the blue lower flat curve is for GLH, red lower volatile curve is for BHZ and green upper flat curve is for the linear exact search.

Figure 2. Repatched images: Three original images are in the first column. Images repatched by GLH are in the second column. Images repatched by BHZ are in the third column.

GLH algorithm performs better than BHZ [3] on all images in terms of quality. Also, Figure 1(b) shows that the GLH algorithm is often faster and more importantly, has significantly less variability in its speed. We note that the number of operations for each algorithm and not actual time is reported in Figure 1(b). In Figure 2, we demonstrate repatching of three images by both GLH and BHZ. In the first column are the original images, in the second column are images repatched by GLH and in the third column images repatched by BHZ. Our algorithm is able to obtain better near neighbors, and hence much better visual quality.

5.2. Face Recognition with Two Databases

Images of faces with fixed pose under different illumination conditions lie near linear subspaces of dimension 9 (Epstein et al. [7], Ho et al. [12], Basri and Jacobs [4]). Therefore a database of faces can be easily transcribed into a set of subspaces (where each subspace represents a face). If a query is a single image of a face, then the problem is to recognize the closest face (subspace) to the given image (point). Alternatively, the query can include several images of the same face under different illumination conditions. In this case, the query is a subspace. Basri et al. [3] used uncropped images which are easier to recognize due to background, clothing and characteristic position of each subject. In the following two experiments, we use cropped images.

We first used cropped images of the Multi-Pie database (see [10]) with frontview. That is, we used all subfolders of the form 05_0 for all persons of all four sessions of the multiview folder. There were a total of 239 persons and 80 frontview images for each. We cropped these well-aligned images by restricting the set of pixels and then used 23×19 cropped images. The subspaces of different faces (under different illumination conditions) are often close to each other (Epstein et al. [7], Ho et al. [12], Basri and Jacobs [4]). In theory, these subspaces are at most 9-dimensional (as established in [4]), however, 9-dimensional subspaces are hard to distinguish due to this proximity. We have experi-

mentally found out that the subspaces are approximately of dimension 5 and thus use this dimension (unlike [3] who use 9).

For each person, we randomly picked 36 23×19 cropped images out of the 80 frontview images, vectorize them to lie in 437 dimensions and recorded their total least squares 5-dimensional subspace (spanned by the top 5 principal components). We created 239 such subspaces (one for each person). For each $d = 1, ..., 10$, we created 239 query subspaces as the span of d randomly picked images from the rest of the 44 images (the ones not used to create the database). The results of the GLH scheme, compared to BHZ scheme, are summarized in Figure 3, where for each query dimension d, the darker bar represents the success rate (among 239 query d-dimensional subspaces) of GLH and the brighter bar represents success rate of BHZ. GLH performs significantly better for $d \geq 4$. We note that GLH takes place in 437 dimensions, while BHZ takes place in 95703 dimensions (437*(437+1)/2=95703).

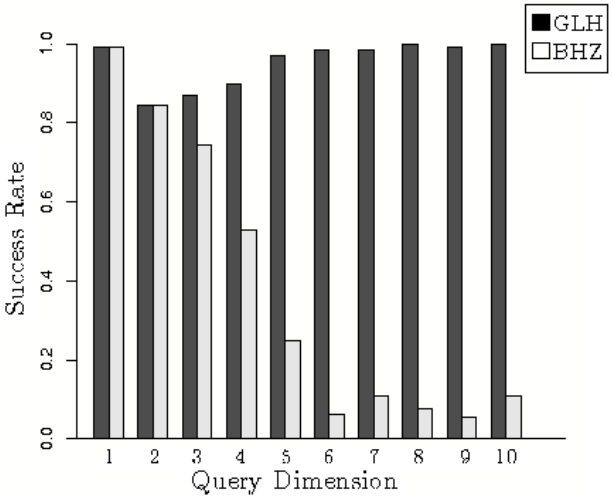


Figure 3. Success Rate for different Query dimension

We performed a similar experiment with the cropped images of the Extended Yale Face Database B [8, 14]. In this database, there are 38 persons, where for each person there are 64 different 24×21 face images with different illuminations. The database of 38 5-dimensional subspaces is created by randomly choosing 36 images out of the 64 images per person, vectorizing them to lie in 504 dimensions and computing their 5-dimensional total least squares subspace. Similarly to the previous experiment, for $d = 1, ..., 10$ we form the query d-dimensional subspaces by the span of randomly chosen d vectors from the other 28 images. Figure 4 report the success rate (among 38 queries) of GLH and BHZ for $1 \leq d \leq 10$. GLH performs significantly better than BHZ across all dimensions d. GLH takes place in 504 dimension, whereas BHZ in 127,260 dimensions (504*505/2=127,260).

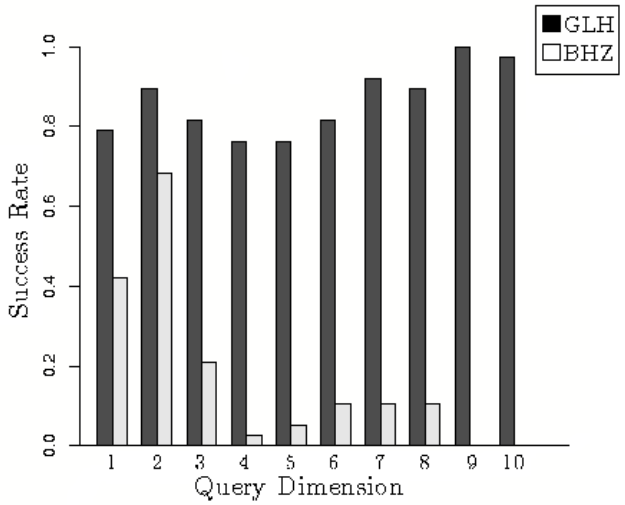


Figure 4. Success Rate for different Query dimension

6. Conclusion

We have proposed GLH, a sublinear time algorithm for the approximate point-to-subspace query and subspace-to-subspace query problems. It is based on a new locality sensitive hashing family, which takes advantage of the geometric position of different subspaces as encoded in their principal angles with random lines. The GLH algorithm performs stably and reliably in numerical experiments, and in particular outperforms previous approaches to approximate nearest subspace.

Although this method provides good results in our experiments, there is still room for improvement. First, it is desirable to extend the method to also handle affine subspaces. This would require hashing families that encode not only angles but also distances and maintain locality sensitivity hashing. Secondly, as with other algorithms for this problem, in extremely high ambient dimension, GLH shows the greatest advantage over linear scan when the database is very large. Therefore, it is desirable to find hashing families that work well even when the database is small.

Acknowledgements. We acknowledge the financial support of NSF and ONR: XW and GL were partially supported by NSF grants DMS-09-15064 and DMS-09-56072 (awarded to GL), JW was partially supported by ONR N00014-13-1-0492, and GL was also partially supported by the IMA (during 2011-2012). XW, SA and GL thank the

DTC and IMA at UMN for their hospitality and encouragement of cross collaborations. Finally, we thank the reviewers for their useful feedback.

7. Appendix

Implementation of GLH and BHZ. For GLH, we chose the following parameters: $K = 3$, $S = 20$ and $\theta = \pi/8$. BHZ is based on ANN (approximate nearest neighbor) search. We used the ANN implementation provided by David M. Mount and Sunil Arya.[3] with the following parameters: split_rule = 'suggest', shrink_rule = 'none', eps = 10, near_neigh = 1, run_queries = 'priority'.

For BHZ, the trade-off between efficiency and quality can be achieved by tuning the parameter eps in the ANN search. We used the same eps that was mentioned in Basri et al. [3] (eps = 10). With this choice of eps, GLH and BHZ had comparable speed in the image approximation task; while in the recognition task, BHZ was slower than GLH (e.g., about 10 times slower for the cropped images of the Extended Yale Face Database B [8, 14]). Smaller values of eps results in more accurate results of BHZ, but with slower speed.

References

[1] A. Andoni and P. Indyk. Near optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Comm. of the ACM*, 51(1), 2008.

[2] A. Andoni, P. Indyk, R. Krauthgamer, and H. L. Nguyen. Approximate line nearest neighbor in high dimensions. *SODA*, 2009.

[3] R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate nearest subspace search. *TPAMI*, 33(2), 2011.

[4] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *TPAMI*, 25(2):218–233, February 2003.

[5] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[6] K. Clarkson. A randomized algorithm for closest-point queries. *SICOMP*, 17:830–847, 1988.

[7] R. Epstein, P. Hallinan, and A. Yuille. 5 ± 2 eigenimages suffice: An empirical investigation of low-dimensional lighting models. In *IEEE PBMCV*, June 1995.

[8] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *TPAMI*, 23(6):643–660, 2001.

[9] G. H. Golub and C. F. Van Loan. Matrix computations, 3rd edition. *Baltimore: Johns Hopkins University Press*, 1996.

[10] R. Gross, I. Matthews, J.F. Cohn, T. Kanade, and S. Baker. Multi-pie. *Proceedings of the Eighth IEEE International Conference on Automatic Face and Gesture Recognition*, 2008.

[11] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8:321–350, 2012.

[12] J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *CVPR*, 2003.

[13] P. Jain, S. Vijayanarasimhan, and K. Grauman. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. In *NIPS*, 2010.

[14] K.C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *TPAMI*, 27(5):684–698, 2005.

[15] W. Liu, J. Wang, Y. Mu, S. Kumar, and S. Chang. Compact hyperplane hashing with bilinear functions. In *International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, 2012.

[16] A. Magen. Dimensionality reductions that preserve volumes and distance to affine spaces, and their algorithmic applications. *RANDOM*, 2002.

[17] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[18] Y. Mu, J. Wright, and S. Chang. Accelerated large scale optimization by concomitant hashing. In *Computer Vision–ECCV*, pages 414–427. Springer, 2012.

[19] J. Sun, Y. Zhang, and J. Wright. Efficient point-to-subspace query in ℓ^1 with application to robust face recognition. 2012.

[20] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.

[21] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[22] A. Yang, J. Wright, Y. Ma, and S. Sastry. Unsupervised segmentation of natural images via lossy data compression. *Comput. Vis. Image Underst.*, 110(2), May 2008.

[23] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. *Image Processing, IEEE Transactions on*, 21(5):2481–2499, 2012.

[3] The ANN library is available on http://www.cs.umd.edu/ mount/ANN/