A General Framework for Approximate Nearest Subspace Search

Ronen Basri*
Weizmann Institute of Science
Rehovot, 76100, Israel

Tal Hassner**
The Open University of Israel
Raanana, 43107, Israel

Lihi Zelnik-Manor
Technion - Israel Inst. of Technology
Haifa, 32000, Israel

Abstract. Subspaces offer convenient means of representing information in many Pattern Recognition, Machine Vision, and Statistical Learning applications. Contrary to the growing popularity of subspace representations, the problem of efficiently searching through large subspace databases has received little attention in the past. In this paper we present a general solution to the Approximate Nearest Subspace search problem. Our solution uniformly handles cases where both query and database elements may differ in dimensionality, where the database contains subspaces of different dimensions, and where the queries themselves may be subspaces. To this end we present a simple mapping from subspaces to points, thus reducing the problem to the well studied Approximate Nearest Neighbor problem on points. We provide theoretical proofs of correctness and error bounds of our construction and demonstrate its capabilities on synthetic and real data. Our experiments indicate that an approximate nearest subspace can be located significantly faster than the nearest subspace, with little loss of accuracy.

1 Introduction

Although the use of subspace representations has increased considerably over the years, one fundamental question related to their use has so far received little attention: How does one efficiently search through a database of subspaces? There are two main reasons why we believe this question to be paramount. The first is the demonstrated utility of subspaces as a (sometimes only) means for conveniently representing varying information. The second is the ever-growing volume of information routinely collected and searched through as part of Computer Vision and Pattern Recognition systems, information often represented by subspaces. The goal of this paper is to address this question by presenting a general framework for efficient subspace search.

In a recent paper [4] Basri et al. have presented a method for sub-linear *approximate nearest subspace* (ANS) search. Their solution, however, was limited to a particular scenario where the queries are high dimensional *points*. They thus ignore cases where the query itself may be a subspace. Moreover, their method cannot handle databases of subspaces with different dimensions, which may be the case if variable amounts of information are available when the database is produced or when object representations allow for different degrees of freedom. In this paper we extend their work and provide the following contributions.

Author names are ordered alphabetically due to equal contribution.

*Part of the work was does while at the Toyota Technical Institute.

**Part of the work was does while at the Weizmann Institute of Science.

- We present a *general* framework for efficient approximate nearest subspace search. Our framework addresses circumstances where both query and database elements may be either points or subspaces of different dimensions. This allows us in particular to handle cases in which the database subspaces are of varying dimensions. This work thus facilitates the use of subspaces, and in particular subspace queries, in a range of applications.
- We rework the math in [4], demonstrating the relation between the Euclidean and the F-norm distance measures, thus obtaining simpler yet more general derivations.
- We provide empirical analysis on both synthetic and real data for the new scenarios handled. In particular, we test the performance of our method on tasks related to illumination, voice, and motion classification.

Because both query and database items may be subspaces, we define their distance as the sum squared sines of the principal angles between them. To efficiently search through a database of subspaces for ones which minimize this distance, we present a simple reduction to the problem of efficient *approximate nearest neighbor* (ANN) search with *point* queries and database elements [1, 2, 19]. We further show that the particular circumstance handled by [4] is a special case of the mapping presented here.

We next survey related work, describe our method including theoretical proofs of correctness and error bounds of our construction, and present both analytical and empirical analysis.

2 Previous Work

The literature on subspace representations is immense and so is the number of applications utilizing them. The popularity of subspaces is due to the observation that a single subspace can capture an infinite range of transformations applied to a single object. For example, only one subspace is required to represent all possible images of a Lambertian object viewed under different illuminations [5, 21]. Similar representations were constructed for objects viewed under changing spatial transformations (e.g. using the "tangent distance" [23]), viewpoint [24, 26], and articulation [10, 11, 25]. Subspaces have additionally been used to represent an object's identity [15, 28], classes of similar objects [3, 8] and more.

Consider for example a typical scenario, where a database of high dimensional subspaces is collected, each one representing different transformations of a certain object. Given a query, this database is searched for the query's nearest (or near) subspaces. Basri et al. [4] presented an efficient search method for the particular case of the query being a high-dimensional point and the database containing subspaces of identical dimensions. Although an important first step, their method is insufficient for the following two reasons. The first is strong evidence that often the queries should and sometimes they *must* be subspaces themselves. In [15], for example, Fitzgibbon and Zisserman showed that for the purpose of face recognition subspace-to-subspace distance is a better measure of similarity than point-to-subspace. A similar result was demonstrated empirically even earlier by [27] for face recognition using video streams. Moreover, when using subspaces to capture motion (e.g., [10, 11, 17, 25]) it is unclear how points can even be used to represent queries; subspaces being the natural representation for

both the database items and the queries. These last examples all demonstrate the second shortcoming of [4], namely, in all these applications the database subspaces might differ in dimensionality, a case not handled by their search method.

We should note that subspace search problems have received considerable attention also in theoretical fields of Computer Science. For example, subspaces have been used to solve the so called "Partial Match" problem on strings [12] and related problems. These problems usually use subspaces to represent binary strings with unknown values. The subspaces they handle are therefore parallel to the world axes and only span two values in each coordinate. As such, they present a special case of the one handled here. In his paper [20] Magen proposed an efficient solution to the nearest subspace search problem by a reduction to the vertical ray shooting problem. However, besides being applicable only to point queries, his solution requires preprocessing time exponential in the subspace dimension and so is impractical in many applications.

3 Nearest Subspace Search

The *nearest subspace search problem* is defined as follows. Let $\{\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^n\}$ be a collection of *linear* (or *affine*) subspaces in $\mathcal{R}^d$, each with intrinsic dimension k_{S^i}. Given a query subspace $\mathcal{Q} \subset \mathcal{R}^d$, with intrinsic dimension k_Q, denote by $\mathrm{dist}(\mathcal{Q}, \mathcal{S}^i)$ a distance measure between the subspaces $\mathcal{Q}$ and $\mathcal{S}^i, 1 \leq i \leq n$. We seek the subspace $\mathcal{S}^*$ that is nearest to $\mathcal{Q}$, i.e., $\mathcal{S}^* = \arg\min_i \mathrm{dist}(\mathcal{Q}, \mathcal{S}^i)$. For notational simplicity we omit below the superscript index and refer to a database subspace as $\mathcal{S}$. The meaning should be clear from the context.

There are many possible definitions of the distance between two *linear* subspaces [13]. Our particular choice of distance will be discussed in the following section. To the best of our knowledge there is no accepted distance measure between affine subspaces. We will thus limit our discussion at this point to the case of linear subspaces. Later on, in Section 3.5 we will revisit the affine subspace case and propose possible solutions.

Following [4] we approach the nearest subspace problem by reducing the problem to the well explored nearest neighbor (NN) search problem for points. To achieve such a reduction we define two transformations, $\mathbf{u} = f(\mathcal{S})$ and $\mathbf{v} = g(\mathcal{Q})$, which respectively map any given database subspace $\mathcal{S}$ and query subspace $\mathcal{Q}$ to points $\mathbf{u}, \mathbf{v} \in \mathcal{R}^{d'}$ for some d', such that the Euclidean distance $\|\mathbf{v} - \mathbf{u}\|_2$ increases monotonically with $\mathrm{dist}(\mathcal{Q}, \mathcal{S})$. In particular, we derive below such mappings for which

$$\|\mathbf{v} - \mathbf{u}\|_2^2 = \mu\, \mathrm{dist}^2(\mathcal{Q}, \mathcal{S}) + \omega \tag{1}$$

for some constants μ and ω.

This form of mapping was was shown [4] to be successful for point queries. Here we start by proposing a simple yet general mapping that can handle both point queries as well as subspace queries, when the database subspaces are all of the same intrinsic dimension (Section 3.1). In Section 3.3 we refine the mapping to obtain better error bounds. Later on, in Section 3.4 we show how this mapping can be extended to handle databases of subspaces of varying dimensions.

3.1 A Simple Reduction to Nearest Neighbor Search

We represent a database linear subspace $\mathcal{S} \subset \mathcal{R}^d$ by a $d \times k_S$ matrix S with orthonormal columns. We represent a point query by a $d \times 1$ vector $\mathbf{q}$ and a subspace query as a $d \times k_Q$ matrix Q with orthonormal columns.

Next, we need to define the distance measure $\mathrm{dist}^2(\mathcal{Q}, \mathcal{S})$ between two subspaces. As was shown in [13], all common distance definitions are based on the principal angles $\boldsymbol{\theta} = (\theta_1, \theta_2, ...)$ and are monotonic with respect to each other. That is, sorting the database subspaces according to their distance from the query subspace will produce the same order, regardless of the distance definition. Therefore, the choice of distance measure is based on its applicability to mappings of the form in Eq. (1). After some investigation, we chose to adopt the projection Frobenius norm defined as $\mathrm{dist}^2(\mathcal{Q}, \mathcal{S}) = \|\sin \boldsymbol{\theta}\|_2^2$, where $\sin \boldsymbol{\theta}$ is the vector of sines of the principal angles between the subspaces $\mathcal{S}$ and $\mathcal{Q}$. When $\mathcal{Q}$ and $\mathcal{S}$ are of the same dimension $k_S = k_Q = k$ the vector $\sin \boldsymbol{\theta}$ is of length k, while when they differ in dimension its length is $k_{\min} = \min(k_S, k_Q)$.

This distance was selected since it has three important properties:

- A linear function of the squared distance can be obtained via the Freobenius norm of the difference between the orthographic projection matrices of the subspaces (aka its name):

$$\begin{aligned} \|QQ^T - SS^T\|_F^2 &= k_Q + k_S - 2\sum_{i=1}^{k_{\min}} \cos^2 \theta_i \\ &= k_Q + k_S - 2k_{\min} + 2\mathrm{dist}^2(\mathcal{Q}, \mathcal{S}). \end{aligned} \tag{2}$$

- We can use the projection F-norm also to compute the distance between a point query $\mathbf{q} \in \mathcal{R}^d$ and a database subspace S, since the Euclidean distance between them, denoted $\mathrm{dist}(\mathbf{q}, \mathcal{S})$, is, up to a linear transformation, equal to the projection F-norm between the 1D space through $\mathbf{q}$ and S:

$$\begin{aligned} \|\mathbf{q}\mathbf{q}^T - SS^T\|_F^2 &= \|\mathbf{q}\mathbf{q}^T\|^2 + \|SS^T\|^2 - 2\mathbf{q}^T SS^T \mathbf{q} \\ &= \|\mathbf{q}\|^4 + k_S - 2\|\mathbf{q}\|^2 + 2\mathrm{dist}^2(\mathbf{q}, \mathcal{S}). \end{aligned} \tag{3}$$

- Finally, we note, that the Frobenius norm of a square matrix A can be computed by summing the squares of all its entries: $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$. This implies that it can also be computed as the L_2 norm of a vector $\mathbf{a}$ such that $\|A\|_F^2 = \|\mathbf{a}\|_2^2$ and $\mathbf{a}$ is a vector containing all entries of A.

These observations imply that a mapping based on rearranging the projection matrices SS^T and QQ^T into vectors could be of the form defined in Eq. (1). Since the projection matrices are symmetric, naïve rearrangement of their entries will result in redundancy. We thus further define the following operator: For a symmetric $d \times d$ matrix A we define an operator $h(A)$, where h rearranges the entries of A into a vector by taking the entries of the upper triangular portion of A, with the diagonal entries scaled by $1/\sqrt{2}$, i.e.,

$$h(A) = (\frac{a_{11}}{\sqrt{2}}, a_{12}, ..., a_{1d}, \frac{a_{22}}{\sqrt{2}}, a_{23}, ..., \frac{a_{dd}}{\sqrt{2}})^T \in \mathcal{R}^{d'} \tag{4}$$

and $d' = d(d+1)/2$. Our generalized mapping can now be defined as follows:

$$\begin{aligned} \mathbf{u} &\doteq f(\mathcal{S}) = h(SS^T) \\ \mathbf{v} &\doteq g(\mathcal{Q}) = h(QQ^T). \end{aligned} \tag{5}$$

This mapping is consistent with the desired distance definition of Eq. (1) with $\mu = 1$ when all database subspaces $\mathcal{S}$ are of the same intrinsic dimension $k_S = k \ \forall S$. The additive constant ω depends on the query. One can show that for subspace queries with $k_Q = k_S = k$ we get $\omega = 0$, while for subspace queries of different dimension $k_Q \neq k_S$ we get $\omega = \frac{1}{2}(k_S + k_Q) - k_{\min}$ which is mutual to all database items, implying a valid mapping. Moreover, this mapping applies to point queries where we get $\omega = \frac{1}{2}\|\mathbf{q}\|^4 - \|\mathbf{q}\|^2 + \frac{1}{2}k$.

Note, that these observations imply that the same mapped database can be utilized for various query types *without* knowing a-priori which queries will be applied. This can be useful in many applications, for example, in face recognition the number of available images can vary depending on application. At times only a single query image will be available, but when the face is captured, for example, via a web-cam many occurrences of it may be available and can be used to fit a linear subspace as was proposed in [27]. The mapping of Eq. (5) allows using a single database for all queries regardless of dimension.

3.2 Is this a Good Mapping?

The quality and speed of the search depend highly on the constants μ and ω. One can show that with mappings of the form in Eq. (1) to guarantee an approximation ratio (error bound) of $1 + E$ in the original distance $r = \mathrm{dist}(\mathcal{Q}, \mathcal{S})$ we would need to select an approximation ratio $1+\epsilon = \left(\frac{\omega/\mu + r^2(1+E)^2}{\omega/\mu + r^2}\right)^{1/2}$ in the search on the mapped points. We would therefore like the ratio ω/μ to be minimized. A large ratio ω/μ means the entire database is pushed away from the query requiring longer search times and using smaller values of ϵ to maintain result quality. The mapping of Eq. (5) is thus "ideal" with $\omega = 0$ for queries of dimension equal to the database subspaces, but not so for queries of a different dimension. Ideally, one would like to eliminate the additive constant ω also for the case of queries of different dimension. Unfortunately, a non-zero additive constant ω is inevitable when the query and database subspaces differ in dimension.

Lemma: Let $\mathcal{S}$ and $\mathcal{Q}$ be subspaces $\in \mathcal{R}^d$ with intrinsic dimensions k_S and k_Q, respectively and $k_S \neq k_Q$. Let $\mathbf{u} = f(\mathcal{S})$ and $\mathbf{v} = g(\mathcal{Q})$, be their mapping into points $\mathbf{u}, \mathbf{v} \in \mathcal{R}^{d'}$ for some d', such that the distance between the mapped points is of the form $\|\mathbf{v} - \mathbf{u}\|_2^2 = \mu\,\mathrm{dist}^2(\mathcal{Q}, \mathcal{S}) + \omega$. Then $\omega \neq 0$.

Proof: When $\mathcal{S} \subset \mathcal{Q}$ or $\mathcal{Q} \subset \mathcal{S}$ then by definition $\mathrm{dist}^2(\mathcal{Q}, \mathcal{S}) = 0$. If $\omega = 0$ we get $u = v$. That is, any two subspaces with non-trivial intersection must be mapped to the same point. Since there exists a chain of intersections between any two subspaces, the only possible mapping in the case that $\omega = 0$ is the trivial mapping. $\diamond$

Note, that this is true for any mapping from subspaces to points and is not limited to the mapping of the form chosen in this paper. While ω cannot be eliminated the ratio ω/μ can be further minimized, as is shown next.

3.3 Improving the Error Bounds

First, we denote by $\boldsymbol{t} = \sqrt{2}h(I_d) \in \mathcal{R}^{d'}$ (I_d denotes the $d \times d$ identity matrix), a vector whose entries are one for each diagonal entry in h(.) and zero elsewhere. Database subspaces mapped using Eq. (5) lie on the intersection of a sphere and a hyperplane; they lie on a sphere since all share the same length $\|\mathbf{u}\|^2 = \frac{1}{2}k_S$, they lie on a hyperplane orthogonal to $\boldsymbol{t}$ because $\boldsymbol{t}^T\mathbf{u} = k_S/\sqrt{2}$ (since the trace of a projection matrix is constant). If the query is of a different intrinsic dimension it will be mapped onto the intersection of different sphere and hyperplane (see Fig. 1). To reduce the distance between the mapped query and the mapped database items we can modify our mapping such that all mapped subspaces lie on the intersection of the *same* hyperplane and the *same* sphere (see Fig. 1). This modification maintains the monotonicity of the mapping.

We implement this modification as follows. We start by modifying our mapping such that the mapped query is projected onto the hyperplane of mapped subspaces. We first translate the hyperplane of mapped database subspaces so that it goes through the origin, by setting $\bar{\mathbf{u}} = \mathbf{u} + \alpha\boldsymbol{t}$ with $\alpha = -k_S/(d\sqrt{2})$. The hyperplane after this translation is given by $\boldsymbol{t}^T\bar{\mathbf{u}} = 0$. Given a query Q and its mapped version $\mathbf{v}$ we seek to project $\mathbf{v}$ onto this translated hyperplane. That is, we seek a scalar β such that $\bar{\mathbf{v}} = \mathbf{v} + \beta\boldsymbol{t}$ lies on the hyperplane $\boldsymbol{t}^T(\mathbf{v} + \beta\boldsymbol{t}) = 0$. Using the identities $\boldsymbol{t}^T\mathbf{v} = k_Q/\sqrt{2}$ and $\boldsymbol{t}^T\boldsymbol{t} = d$ we obtain $\beta = -k_Q/(d\sqrt{2})$.

Next, we wish to uniformly scale the query to bring it to the same sphere as the database items. Such uniform scaling too maintains the monotonicity of the mapping. To simplify notations, we scale both database items and the query to have unit norm. Our final mapping is as follows,

$$\begin{aligned}\mathbf{u} \doteq f(\mathcal{S}) &= \frac{1}{c_S}\left(h(SS^T) - \frac{k_S}{d\sqrt{2}}\boldsymbol{t}\right)\\ \mathbf{v} \doteq g(\mathcal{Q}) &= \frac{1}{c_Q}\left(h(QQ^T) - \frac{k_Q}{d\sqrt{2}}\boldsymbol{t}\right),\end{aligned} \tag{6}$$

with $c_S = \sqrt{\frac{1}{2}k_S(1 - k_S/d)}$ and $c_Q = \sqrt{\frac{1}{2}k_Q(1 - k_Q/d)}$. This mapping implies $\|\mathbf{v} - \mathbf{u}\|_2^2 = \mu\,\mathrm{dist}^2(Q, S) + \omega$, where $\mu = \frac{1}{c_S c_Q}$ and $\omega = 2 - \frac{k_{\min}}{c_S c_Q} + \frac{k_S k_Q}{d c_S c_Q}$.

The constants $\mu > 0$ and $\omega \geq 0$ depend only on k_S, k_Q and d and are thus both mutual to all database items and maintain monotonicity with respect to the true distance between subspaces. When the query and database have equal intrinsic dimensions, i.e., $k_S = k_Q$, we get $\mu = 2d/(kd - k^2)$ and $\omega = 0$ implying that the mapping of Eq. (6) reduces to the mapping of Eq. (5), up to a scale factor. When the intrinsic dimension of the query and database subspaces are significantly smaller than the ambient space dimension, i.e., $k_S, k_Q \ll d$ we get $\mu \approx 2/\sqrt{k_S k_Q}$ and $\omega \approx 2(1 - k_{\min}/\sqrt{k_S k_Q})$.

3.4 Subspaces of Varying Dimension

In some applications the database itself can contain subspaces of varying dimension. This may be the case, for example, when the database contains visual descriptions of different articulated objects with varying degrees of freedom. It could also arise in face

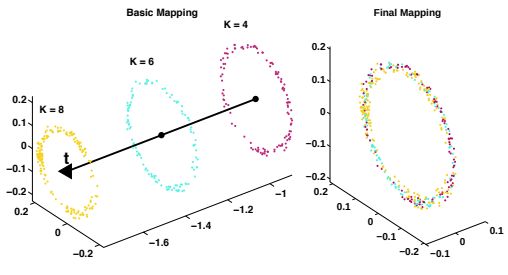

Fig. 1. The geometry of the mapped subspaces. Left: Slicing through mapped subspaces of intrinsic dimensions 4,6 and 8 in a 10 dimensional space, shows that the basic mapping of Eq. (5) maps subspaces of different dimensions onto different intersections of spheres and hyperplanes. Right: The refined mapping of Eq. (6) aligns the spheres.

recognition when varying number of images (from one to many) are available for different faces. The mapping of Eq. (5) cannot be used in such scenarios since it implies that ω depends on k_S and is thus *not* mutual to all database items, breaking the monotonicity. Next, we propose mappings that remove the dependence on the database subspace dimension, thus allowing handling within a single framework databases with subspaces of varying intrinsic dimensions.

$\boldsymbol{k_Q > k_S, \forall S}$: When the intrinsic dimension of the query subspace is *larger* than that of *all* database subspaces, i.e., $k_Q > k_S \; \forall S$, we can modify the mapping so that it does not depend on k_S.

$$\begin{aligned} \mathbf{u} &\doteq f(\mathcal{S}) = h(SS^T) \\ \mathbf{v} &\doteq g(\mathcal{Q}) = \frac{1}{2}h(QQ^T), \end{aligned} \tag{7}$$

and consequently $\|\mathbf{v}-\mathbf{u}\|_2^2 = \frac{1}{8}k_Q + \frac{1}{2}k_S - \frac{1}{2}\|\cos\boldsymbol{\theta}\|^2 = \frac{1}{8}k_Q + \frac{1}{2}\mathrm{dist}^2(\mathcal{Q},\mathcal{S})$. This implies we have obtained a mapping for which $\|\mathbf{v}-\mathbf{u}\|_2^2 = \mu \; \mathrm{dist}^2(\mathcal{Q},\mathcal{S}) + \omega$, where $\mu = \frac{1}{2}$ and $\omega = \frac{1}{8}k_Q$. This distance is independent of the database subspace dimension k_S and thus the mapping of Eq. (7) can be used for all subspaces even when their intrinsic dimensions vary.

$\boldsymbol{k_Q \leq k_S, \forall S}$: When the intrinsic dimension of the query subspace is *smaller* than that of *all* database subspaces, i.e., $k_Q \leq k_S \; \forall S$, we can eliminate the dependence on k_S by introducing an additional entry to the mapped subspaces, as follows.

$$\begin{aligned} \mathbf{u} &\doteq f(\mathcal{S}) = [h(SS^T)\;,\; \sqrt{0.5(k_{\max} - k_S)}] \\ \mathbf{v} &\doteq g(\mathcal{Q}) = [h(QQ^T)\;,\; 0], \end{aligned} \tag{8}$$

where $k_{\max} = \max_S k_S$. Consequently, $\|\mathbf{v}-\mathbf{u}\|_2^2 = \mu \; \mathrm{dist}^2(\mathcal{Q},\mathcal{S}) + \omega$, where $\mu = 1$ and $\omega = \frac{1}{2}k_{\max} - \frac{1}{2}k_Q$. Hence, Eq. (8) provides a valid mapping such that the distance between mapped subspaces is a linear function of the true distance with constants mutual to all database items.

Arbitrary query dimension. This leaves us with the case that the query dimension is smaller than the dimension of some elements in the database and larger than others. Unfortunately, we cannot obtain a single mapping in which the distance between the mapped subspaces is independent of k_S when k_S is free to be larger or smaller than k_Q. The reason for this is that the true distance between two subspaces is obtained

by summing over $k_{\min} = \min(k_S, k_Q)$ angles and thus this distance depends on the relation between k_Q and k_S.

When the query dimension k_Q is known a-priori and fixed for all queries, we propose to pre-process the database twice, once with the mapping of Eq. (7), which is appropriate for database subspaces with $k_Q > k_S$ and once with the mapping of Eq. (8), which is appropriate for database subspaces with $k_Q \leq k_S$. We apply each mapping only to the appropriate portion of the database. Given a query, we perform a search in each of the two mapped databases. From each search we obtain a candidate nearest neighbor, compute the true distance to the two and select the closer one. This does not modify the pre-processing time and memory requirement, but it does make running time slightly slower. Still, it is much faster than full linear search (see Sec. 4).

When k_Q is unknown a-priori and can vary we apply each mapping to the entire database. Given a query, we search both of the mapped databases. From each search we obtain an a-priori chosen number of candidate nearest neighbors and compute the true distance to those that are appropriate for the corresponding mapping. We then select the nearest neighbor out of those. This doubles the pre-processing time, running time and memory requirement, but is still faster than full linear search. A problem with this approach is that since each of the two mappings is appropriate for only part of the database subspaces, we cannot guarantee that the extracted candidate near neighbors are appropriate and consequently we cannot guarantee bounds on the error.

An alternative solution is to pre-process the database for all possible values of k_Q. For each possible k_Q we split the database into two subsets, one including all subspaces $k_S \geq k_Q$ and the other with all subspaces such that $k_S < k_Q$. Given a query we proceed as above, searching the two appropriate pre-processed databases. Note that at most we need to pre-process the database for $\max(k_S) - \min(k_S) + 1$ values of k_Q. This increases the pre-processing time and memory requirement by a factor of $(\max(k_S) - \min(k_S) + 1)$, however, running time is still only doubled. Yet another alternative is to create $\max(k_S) - \min(k_S) + 1$ mapped databases, each including only subspaces of one possible value of k_S. The runtime in such a solution would be slower since we will need to search $\max(k_S) - \min(k_S) + 1$ databases instead of two, however, the required space would be significantly smaller as each database subspace is mapped and stored only once.

Refining the mapping as was proposed in Eq. (6), is impossible in this case since the database subspaces are mapped onto different hyperplanes, depending on k_S.

3.5 Affine Subspaces

As far as we know, there is no accepted distance measure between affine subspaces. This is probably since affine subspaces are defined by two different components with different natures: a linear part, defining the subspace orientation, and an offset vector from the origin. The distance between two affine subspaces can depend on both the difference in orientation and the difference in offset, however, these distances are not defined in the same units, one is an angular difference while the other is a length, and thus cannot be easily combined into a single unified metric.

A possible approach to incorporating the angular distance and the offset distance is, given an affine subspace $\mathcal{A}$ of intrinsic dimension k_A in R^d, to embed $\mathcal{A}$ as a linear

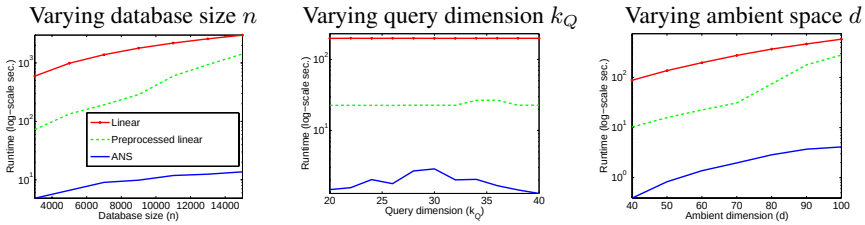

Fig. 2. Synthetic data tests. log-scale run times compared for an exact linear search, linear search with a preprocessed database and our ANS method with ANN $\epsilon = 100$. The following tests were performed. Varying n: Database subspaces of $k_S = 30$ embedded in $d = 60$ space, tested with 1000 queries of dimension $k_Q = 10$. Varying k_Q: Database contains 1000 subspaces with $k_S = 30$, tested with 1000 queries. Varying d: Database contains 1000 subspaces with $k_S = 30$, tested with 1000 queries of dimension $k_Q = 10$. Err rate for our method remained at an almost constant 0.01 in all three experiments, through all values tested.

subspace in R^{d+1} with intrinsic dimension $k_A + 1$. The distance between two affine subspaces is then defined as the distance between the corresponding higher-dimensional embedded linear subspaces. This is inspired by the computation of distance between optical flow directions, proposed in [6]. Having reduced the problem to that of linear subspaces we can use the appropriate mapping as proposed in the previous sections.

4 Complexities

Given a query, our search routine starts by mapping it to $O(d^2)$-space using Eqs. (5) or (6) (or (7),(8) in case of a database with subspaces of varying dimension), and then searching for an approximate nearest neighbor using a point based method (e.g. [2, 1]). Mapping the query requires $O(k_Q d^2)$ time, giving us the following general expression for the query runtime: $O(k_Q d^2) + T_{ANN}(n, d^2)$, where $T_{ANN}(n, d^2)$ is the running time for a choice of an ANN algorithm, on a database of n points in $\mathcal{R}^{d^2}$.

One ANN method is the search-tree based approaches (e.g. [2]). Given an acceptable error rate $\epsilon > 0$ these methods report a point whose distance from the query is at most a $(1 + \epsilon)$-factor larger from the distance of the nearest point from the query. Their runtime is $T_{ANN}(n, d) = O(d^{d+1}\epsilon^{-d} \log n)$. Despite the exponential term these methods tend to run much faster than a sequential scan even in fairly high dimensions.

An alternative approach for ANN is the Locality Sensitive Hashing (LSH) scheme (e.g., [1]), designed to solve the *near neighbor* problem. Given r and ϵ these methods seek a neighbor of distance at most $r(1 + \epsilon)$ from the query, providing that the nearest neighbor lies within distance r from the query. LSH finds a near neighbor in $O(dn^{1/(1+\epsilon)^2+O(1)})$ operations. An ANN can then be found using an additional binary search on r, increasing the overall runtime complexity by a $O(\log n/\epsilon)$ factor.

The preprocessing time includes applying our mapping to the n database subspaces, and then inserting them into a search structure, giving us: $O(nk_S d^2) + T_{pANN}(n, d^2)$, with $T_{pANN}(n, d^2)$ being the preprocessing running time for a choice of an ANN algorithm. For the LSH scheme [1], for example, $T_{pANN}(n, d^2) = O(d^2 n^{1+1/(1+\epsilon)^2+O(1)})$,

depending on the acceptable ϵ error rate, while for the kd-tree scheme [2] this value is $O(d^2 n \log n)$. Finally, the space required by our method depends on the ANN method used and is, e.g., $O(nd^2)$ for the kd-tree scheme of [2], used in our experiments.

Note, that an exact sequential search for a nearest subspace using the distance $||QQ^T - SS^T||_F^2$, requires $O(k_Q d^2 + n k_S d^2)$. Of course, computing SS^T can be performed at preprocessing, resulting in a query runtime complexity of $O(k_Q d^2) + O(d^2 n)$ (see Fig. 2 for empirical evaluations). We therefore obtain that the runtime difference between our method and a linear search is the difference between an exact and approximate point search on points in $O(d^2)$. The exact search can alternatively be performed by computing $SVD(S^T Q)$ to obtain the cosines of the angles between the query and each database subspace. The complexity of this method is $O(k_Q k_S d + \max(k_Q, k_S)^3)$ and is therefore preferable when both $k_Q, k_S \ll d$. Finally, we note that when the given subspaces lie in high dimension we can use a number of random projections of the subspaces [20] or their mapped versions [18] onto an $O(\log n)$ dimension to reduce complexity while sacrificing some accuracy.

5 Experiments

To evaluate the performance of our ANS scheme we adopt the conventional tests in the field (e.g., [1, 2]). These are based on synthetic data with varying parameters as this is the only way to evaluate asymptotic behavior empirically. In addition to these tests, we further demonstrate the applicability of our ANS search method to a number of real applications. Our experiments show that the ANS scheme can indeed significantly expedite the search for a nearest subspace, with only a small penalty in accuracy. Our implementation is in C and uses the ANN kd-tree code of [2]. OpenCV was used for all our matrix routines.

Synthetic data. We tested our ANS scheme on data sets containing thousands of synthetically produced queries and database elements, of fairly large dimensions (Fig. 2). The tests compare our ANS scheme to a linear search, and a linear search with a preprocessed database (see Sec. 4). Run-times for linear search using SVD were significantly slower than the ones reported here, and so are not displayed. Note that we use an ANN $\epsilon = 100$ as a stand-in for the value of "infinity", that is, "the fastest, least exact search". For stability, tests were performed three times and the median result is reported. Both subspaces and queries were randomly selected from a uniform distribution.

We report for each test its running time and its effective distance error [2, 19], defined as $Err = (1/n_Q) \sum_Q (Dist'/Dist^* - 1)$, where n_Q is the number of queries, $Dist'$ is the distance from query $\mathcal{Q}$ to the subspace selected by our algorithm, and $Dist^*$ is the distance between $\mathcal{Q}$ and its true nearest subspace, computed off line. Our tests demonstrate that the ANS scheme is significantly faster than both linear search methods. In addition, in all our tests, the Err values measured were fairly constant, maintaining the low rate of 0.01. The fact that close matches can be recovered even considering the high value for ϵ, is a well documented property of the kd-trees method.

Scene classification. We next test our method on real image data using the scene classification data of [14]. We randomly selected 10 "training" images and 10 (different) "testing" images of three categories. 50 random coordinates were selected in each

Scene classification

Method	*Run time*	*Correct*
ANS (our result)	2.3 sec.	63%
Exact with preprocessing	13.8 sec.	63%
Exact nearest *patch*	135 sec.	55%

Speaker recognition

Method	*Run time*	*Correct*
ANS (our result)	4.6 sec.	100%
Exact with preprocessing	44.5 sec.	100%
Exact nearest *subspace*	232 sec.	100%

Fig. 3. ANS is approximately an order of magnitude faster with no loss of accuracy.

image. Then, 9 different, overlapping 9×9 patches around each coordinate were used to produce a $k = 5$ subspace by taking their 5 principal components. Subspaces originating from "training" images were stored in our subspace database and those from "testing" images were used as queries.

For each query subspace the database was searched for the nearest neighbor providing the category it originated from. For each "testing" image we counted the number of nearest neighbors originating from each category and adopted the maximum as the class label for that image. Figure 3 compares running time and classification results of our method and exact linear search. It shows that while classification results are comparable our method is almost an order of magnitude faster. In addition, all the extracted patches were stored for a point (patch) database and query. Patch results were inferior, probably since subspaces are a richer and more invariant representation of appearance.

Speaker recognition. We tested our method on voice data from [9], consisting of 31 subjects uttering the same short phrase (2-3 seconds long), three times over a phone connection (a total of 93 samples). Each sample was represented by standard mel-frequency cepstrum frame descriptors for time-frames of 25msec, with overlaps of 50%. One sample per subject was taken for the query set and the other two were used to produce the database. We produced both queries and subspaces in the same manner. The concatenated descriptors of 3 consecutive time-frames were taken as points. 20 such points, overlapping by two time-frames, were used to produce a single linear subspace of dimension $k_Q = k_S = 13$. Each sample thus contributed 20–30 such subspaces for a total of 1280 database and 647 query subspaces.

We search the database for an item to match each query. Each selected database item votes for the identity of the query's subject. The speaker is identified based on the majority vote of the queries from each sample. We ran approximate and linear search with and without preprocessing the database. In all cases recognition rate was 100%, however, even on such a small database, our ANS search was an order of magnitude faster (see Figure 3). To simulate cases of partial query data, we ran these tests again, with $k_Q = 6, 7, 8$ and 9. Results remained similar, except for an occasional single recognition error made by the ANS algorithm. Note, that similar recognition rates were reported on the same data set in [9].

Yale-B face recognition. Subspaces are commonly used to capture the appearance of faces under varying illuminations in recognition systems (e.g., [5, 16]). Here, we test the performance of our approach on a similar application using image data from the Yale-B face data set [16] (see Fig. 5 for example images).

For every subject and pose combination in the Yale-B data set, we randomly chose 18 illuminations as database examples, and fit them with a subspace of dimension $k_S = 9$. Since the information available at query time may vary, we tested query sub-

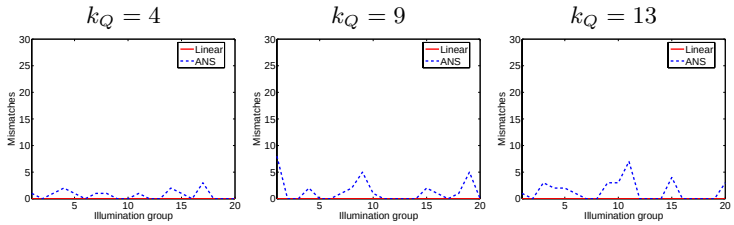

Fig. 4. Yale-B face recognition. Subject mismatches on the Yale-B database [16]. A database of 90 subspaces was produced by fitting subspaces with $k_S = 9$ to 18 randomly selected illuminations for each subject+pose combination. 90 queries were likewise produced by fitting subspaces of $k_Q = 4, 9$, and 13 to randomly selected sets of 4,9, or 13 images from the remaining illuminations. x-axis corresponds to different illumination selections. Recognition rate for our method is 99.2%, 98.4%, and 98.4%, for $k_Q = 4, 9$, and 13.

Fig. 5. Yale-B faces. Example images from the Yale-B data set [16].

spaces of dimensions 4, 9, and 13. For each test we randomly select sets of 4, 9, or 13 images, not used for the database, and used them to fit the query subspaces. Our goal is to recognize the correct face under these conditions. Fig. 4 reports our success rate compared to an exact search. With only 90 subject+pose combinations, our database is far too small to give our method a running time advantage. Still, our ANS method correctly recognized the face at 99.2%, 98.4%, and 98.4%, for $k_Q = 4, 9$, and 13, respectively. These results imply that the performance of our method is mostly influenced by the particular illuminations used to produce the database and queries, and not by the dimension of the query subspaces. For all our tests we used an ANN $\epsilon = 10$. Better results can be obtained, at the price of slower processing speeds, for lower ϵ values.

Motion-based action recognition. Given a video sequence of a person performing an action, we try to classify the action based on example videos of other individuals. Our motion-based similarity measure is motivated by the work of Shechtman and Irani [22]. Their method represents a small space-time (ST) patch P by a matrix $\mathbf{G_P} = [\mathbf{P_x}, \mathbf{P_y}, \mathbf{P_t}]$, where $\mathbf{P_x}$, $\mathbf{P_y}$, and $\mathbf{P_t}$ are column vectors containing the x, y, and temporal gradients for the pixels of P. Two ST patches P and Q are assumed to represent the same motion if they essentially span the same 3-dimensional subspace.

In our tests, we use the action database of [7] (see example frames in Fig. 5), containing ten actions performed by the same nine individuals. For each video sequence, we extract $7 \times 7 \times 3$ patches around a random selection of ST-pixels with a temporal gradient higher than a constant threshold. We use these to produce the matrices $\mathbf{G_P}$ which we then orthonormalize using SVD, for a total of 3200 database subspaces (approximately 40 subspaces per database video sequence). Given a query video of an as yet never seen subject, we similarly construct 500 matrices $\mathbf{G_Q}$ for a random selection

Fig. 6. Motion-based action recognition. Sample frames (cropped) from the action database of [7] used in our action recognition tests.

of its pixels. We then find for each of the query's 3D subspaces, represented by the matrices $\mathbf{G_Q}$, the nearest database subspace $\mathbf{G_P}$. Here again, the final action label is determined by taking the majority vote over the selected database labels.

We compared exact linear subspace search with our approximate subspace search method. Both search methods obtained the same classification rate of 78.9% (although different mistakes were made by each method). More importantly, the mean running time for the linear search was 140 seconds whereas 99 seconds on average were required for the approximate search. We expect this running time advantage to grow with larger databases and careful selection of ANN parameters. Note that although [7] report better classification results, their method takes advantage of additional information. In particular, unlike their method, ours was applied to the unsegmented raw data.

6 Conclusions

We have presented a sub-linear, approximate nearest subspace search method. A single general mapping from subspaces to points was described (with various optimizations), allowing both query and database items to be subspaces, the query to be of a different dimension than the database items, and the database items themselves to vary in dimensions. Once mapped, standard ANN methods can be used to efficiently search the database for nearest neighbors. We believe this method useful for a wide range of applications, and indeed demonstrated its capabilities in a range of experiments. We now plan both to further improve the quality of our mapping to obtain faster search times, and in addition explore various additional problems where it might be applied.

References

1. A. Andoni and P. Indyk. "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *FOCS*: 459–468, 2006.
2. S. Arya, D. Mount, N. Netanyahu, R. Silverman, A. Wu. "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," *Journal of the ACM*, **45**(6): 891–923, 1998. Source code available from www.cs.umd.edu/∼mount/ANN/.
3. J.J Atick, P.A. Griffin, A.N. Redlich, "Statistical Approach to Shape from Shading: Reconstruction of Three-Dimensional Face Surfaces from Single Two- Dimensional Images," *Neural Computation*, **8**(6): 1321-1340, 1996.
4. R. Basri, T. Hassner and L. Zelnik-Manor, "Approximate Nearest Subspace Search with Applications to Pattern Recognition," *CVPR*, in print, 2007.
5. R. Basri, D. Jacobs, "Lambertian reflectances and linear subspaces," *IEEE TPAMI*, **25**(2): 218–233, 2003.
6. J.L. Barron, D.J Fleet , and S.S. Beauchemin, "Performance of optical flow techniques", *IJCV*, **12**(1): 43–77, 1994.

7. M. Blank, L. Gorelick, E. Shechtman, M. Irani, R. Basri, "Actions as Space-Time Shapes," *ICCV*: 1395–1402, 2005. Database available from www.cs.weizmann.ac.il/∼vision/SpaceTimeActions.html.
8. V. Blanz, T. Vetter, "Face Recognition based on Fitting a 3D Morphable Model," *TPAMI*, **25**(9): 1063–1074, 2003.
9. O. Boiman, M. Irani, "Similarity by Composition," *NIPS*,**19**: 177–184, 2006.
10. M.E. Brand, "Morphable 3D models from video," *CVPR*,**2**: 456–463, 2001.
11. C. Bregler, A. Hertzmann and H. Biermann,"Recovering Non-Rigid 3D Shape from Image Streams," *CVPR*, **2**: 690–696, 2000.
12. M. Charikar, P. Indyk and R. Panigrahy, "New Algorithms for Subset Query, Partial Match, Orthogonal Range Searching, and Related Problems," *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, 451–462, 2002.
13. A. Edelman, T.A. Arias and S.T. Smith, "The Geometry of Algorithms with Orthogonality Constraints," *SIAM J. Matrix Anal. Appl.*,**20(2)**: 303–353, 1999.
14. L. Fei-Fei and P. Perona. "A Bayesian Hierarchical Model for Learning Natural Scene Categories," *IEEE Comp. Vis. Patt. Recog.* 2005.
15. A. Fitzgibbon and A. Zisserman, "Joint Manifold Distance: a new approach to appearance based clustering," *CVPR*, **1**: 26–33, 2003.
16. 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," *IEEE TPAMI*, **23(6)**: 643–660, 2001.
17. M. Irani and P. Anandan," Factorization with Uncertainty," *ECCV*,**1**:539–553, 2000.
18. W. Johnson, J. Lindenstrauss, "Extensions of Lipschhitz maps into a Hilbert space," *Contemporary Math*: 189–206, **26**, 1984.
19. T. Liu, A.W. Moore, A. Gray, K. Yang, "An Investigation of Practical Approximate Nearest Neighbor Algorithms," *NIPS*: 825–832, 2004.
20. A. Magen, "Dimensionality reductions that preserve volumes and distance to ane spaces, and their algorithmic applications", *Randomization and approximation techniques in computer science. Lecture Notes in Comput. Sci.,* **2483**: 239–253, 2002.
21. R. Ramamoorthi, P. Hanrahan, "On the relationship between radiance and irradiance: determining the illumination from images of convex Lambertian object." *Journal of the Optical Society of America*, **18**(10): 2448–2459, 2001.
22. E. Shechtman and M. Irani, "Space-Time Behavior Based Correlation OR How to tell if two underlying motion fields are similar without computing them?," *IEEE TPAMI*, To appear, 2007.
23. P. Simard, Y. LeCun, J. Denker and B. Victorri, "Transformation Invariance in Pattern Recognition, Tangent Distance and Tangent Propagation," *Neural Networks: Tricks of the trade*: 227–239, 1998.
24. C. Tomasi, T. Kanade, "Shape and Motion from Image Streams under Orthography: A Factorization Method," *IJCV*, **9**(2): 137–154, 1992.
25. L. Torresani, D. Yang, G. Alexander, C. Bregler, "Tracking and Modeling Non-Rigid Objects with Rank Constraints," *CVPR*: 493–500, 2001.
26. S. Ullman, R. Basri, "Recognition by Linear Combinations of Models," *TPAMI*, **13**(10): 992–1007, 1991.
27. O. Yamaguchi, K. Fukui, and K. Maeda, "Face recognition using temporal image sequence," *Proc. of the 3rd. International Conference on Face and Gesture Recognition*:318–323 1998.
28. H. Zhang, A.C. Berg, M. Maire and J. Malik, "SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition," *CVPR*: **2**:2126–2136, 2006.