Approximate Nearest Subspace Search

Ronen Basri, *Member*, *IEEE*, Tal Hassner, and Lihi Zelnik-Manor, *Member*, *IEEE*

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 problem of Approximate Nearest Subspace search. Our solution uniformly handles cases where the queries are points or subspaces, where query and database elements differ in dimensionality, and where the database contains subspaces of different dimensions. 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.

Index Terms—Approximate nearest neighbor search techniques, subspace representations.

1 Introduction

Linear and affine subspaces are a common means of representing information in computer vision and pattern recognition applications. In computer vision, for example, subspaces are often used to capture the appearance of objects under different lighting [7], [29], viewpoint [35], [33], spatial transformations (e.g., using the "tangent distance" [32]), articulation [13], [34], identity [19], [39], classes of similar objects [4], [10], and more. Typically, given a query image (or images) of an object, represented as a point (or as a subspace) in high-dimensional space, a database of subspaces is searched for the subspace closest to the query. A natural problem which arises from this type of search problems is: Can the nearest (or a near) subspace be found faster than a brute force sequential search through the entire database?

The related problem of finding the nearest neighbor within a database of high-dimensional *point*s has become an important component in a wide range of machine vision and pattern recognition applications. As such, it has attracted considerable attention in recent years, and a number of efficient algorithms for *approximate nearest neighbor* (ANN) search have been proposed (e.g., [3], [16], [22], [25]). These algorithms achieve sublinear search times when locating a near, not necessarily the nearest neighbor, suffices. The gain in query speed is achieved at the price of preprocessing the database. This pays off when a fixed database is searched many times [30]. In light of the success of ANN methods, our goal is to design an *approximate nearest subspace* (ANS) algorithm for efficient search through a database of subspaces.

We present an ANS algorithm based on a reduction to the problem of point ANN search. Our algorithm can thus work in concert with any ANN method, enjoying future improvements to these algorithms. For example, for a query subspace of dimension k_Q and a database of n subspaces (possibly of a different dimension), all embedded in $\mathcal{R}^d$, ANS query running time, using our construction, is $O(k_Q d^2) + T_{ANN}(n, d^2)$, where $T_{ANN}(n, d)$ is the running time for a choice of an ANN algorithm, on a database of n points in $\mathcal{R}^d$. We can achieve further speedup by using random projections to lower the dimensionality of the problem. Our solution uniformly handles cases where the queries are points or subspaces, where query and database elements differ in dimensionality, and where the database contains subspaces of different dimensions.

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

An intuitive approach to the Nearest Subspace problem would be adapting existing solutions designed for the case of points to the case of subspaces. Unfortunately, such extensions are not trivial. Tree-based methods, e.g., [3], [25], rely on splitting the ambient space into smaller cells bounded by hyperplanes. Such a construction cannot be used when the database items are subspaces since they extend infinitely and therefore cannot be bounded in cells. Other methods, e.g., [16], [22], are based on randomly projecting points onto extremely low-dimensional hyperplanes (e.g., lines) and hashing the projected points. The projection of a subspace remains a subspace, and therefore quick hashing remains a problem also on the low-dimensional projection.

- *R. Basri is with the Department of Computer Science and Applied Mathematics, Ziskind Building, Room 227, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: ronen.basri@weizmann.ac.il.*
- *T. Hassner is with the Computer Science Division, The Open University of Israel, 108 Ravutski Str., POB 808, Raanana 43107, Israel. E-mail: hassner@openu.ac.il.*
- *L. Zelnik-Manor is with the Department of Electrical Engineering, Meyer Building, Room 959, The Technion–Israel Institute of Technology, Haifa 32000, Israel. E-mail: lihi@ee.technion.ac.il.*

Manuscript received 30 Sept. 2009; revised 27 Mar. 2010; accepted 5 Apr. 2010; published online 25 May 2010.
Recommended for acceptance by S. Belongie.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-09-0654.
Digital Object Identifier no. 10.1109/TPAMI.2010.110.

TABLE 1
Summary of the Different Combinations of Query and Database Entities Handled by Our Method

		Database			
		Points	Linear subspaces of fixed dimension	Linear subspaces of varying dimension	Affine subspaces
Query	Points	Handled directly by existing ANN methods	√	√	√
	Linear subspaces of fixed dimension	√	√	√	×
	Linear subspaces of varying dimension	√	√	√	×
	Affine subspaces	√	×	×	×

Subspaces have been used to solve the so-called "Partial Match" problem on strings [15] and related problems. These problems usually use subspaces to represent binary strings containing some unknown values. The subspaces they handle are therefore parallel to the main axes and only span two values in each coordinate. As such, they present a special case of the one handled here.

Our method is more related to recent work by Magen [26], who solved ANS by a reduction to the vertical ray shooting problem. Magen's method, however, requires $O(n^{d^2})$ preprocessing time and space, while our preprocessing requires only $O(nk_Sd^2) + T_{pANN}(n, d^2)$ (where k_S is the intrinsic dimensionality of the database subspaces and $T_{pANN}(n, d)$ is the processing time required to produce a search structure for n points of dimension d using, e.g., kd-trees). Furthermore, Magen's approach is applicable only to point queries, while our approach handles both point queries and subspace queries. Finally, a restricted solution to the dual problem has been proposed in [2], where the database consists of points and the query is restricted to be a line.

In an earlier version [5], we presented a method for sublinear *Approximate Nearest Subspace* search which was based on a reduction to the problem of point ANN search. This solution, however, was limited to the particular scenario where the queries are high-dimensional *points*. Although the case of point queries is very often useful, in some cases subspace queries are preferable. For example, in [37], [19], [36], it was shown that, for the purpose of face recognition, subspace-to-subspace distance is a better measure of similarity than point-to-subspace. Moreover, when using subspaces to capture motion (e.g., [13], [14], [23], [34]), 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.

In this paper (see also [6]), we extend and subsume the work of Basri et al. [5] by providing the following contributions:

- 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. The various scenarios handled by our approach are summarized in Table 1.
- We rework the math in [5], 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.

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 (database) of *linear* (or *affine*) subspaces in $\mathcal{R}^d$, each with intrinsic dimension k_{S^i}. Given a query item $\mathcal{Q}$ in $\mathcal{R}^d$, which can be either a point or a subspace with intrinsic dimension k_Q, denote by $\mathrm{dist}(\mathcal{Q}, \mathcal{S}^i)$ the distance between the query item and $\mathcal{S}^i$, $1 \leq i \leq n$. We seek the subspace $\mathcal{S}^*$ that is nearest to the query, 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.

When the query is a point q and $\mathcal{S}$ is a linear or affine subspace, $\mathrm{dist}(q, \mathcal{S})$ is defined as the euclidean distance between them. When the query is a subspace, the distance definition is less straightforward. For example, there are many possible definitions of the distance between two *linear* subspaces [17]. Our particular choice of distance for linear subspaces will be discussed in the following section. At this point, we limit our discussion to the case of linear subspaces. Later on, in Section 4.2, we will revisit the affine subspace case and propose possible solutions. The case of subspace queries with a data set of points is covered in Section 4.2.1.

Following [5], 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 item $\mathcal{Q}$ 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 shown in [5] 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 4.1, 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 linear subspace $\mathcal{S} \in \mathcal{R}^d$ in the database 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 [17], all common distance definitions are based on the principal angles $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots)$ 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 (1). After some investigation, we chose to adopt the Projection Frobenius Norm (Projection F-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, whereas 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 Frobenius norm of the difference between the orthographic projection matrices of the subspaces (hence 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 also use the projection F-norm to compute the distance between a point query $\mathbf{q} \in \mathcal{R}^d$ and a database subspace S since the squared euclidean distance between them, denoted $\mathrm{dist}^2(\mathbf{q},\mathcal{S})$, is, up to a linear transformation, equal to the squared projection F-norm between the 1D space through $\mathbf{q}$ and S:

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

- Finally, we note that the Frobenius norm of a square matrix A can be computed by summing the squares of all of 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 (1). Since the projection matrices are symmetric, naive 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.,

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

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

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

and $\|\mathbf{u}-\mathbf{v}\|^2 = (1/2)\|QQ^T - SS^T\|^2$.

This mapping is consistent with the desired distance definition of (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 (represented as a point) will be available, but when the face is captured, for example, via a Webcam, many occurrences of it may be available and can be used to fit a linear subspace, as was proposed in [37]. The mapping of (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 (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 that 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 (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

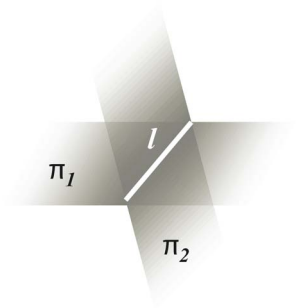


Fig. 1. Graphic illustration of the proof. Subspace l is in the intersection of two higher dimensional subspaces π_1 and π_2. If $\omega = 0$, then all three subspaces will be mapped to the same point.

additive constant ω also for the case of queries of different dimension. Unfortunately, a nonzero additive constant ω is inevitable when the query and database subspaces differ in dimension. We prove this via the following lemma:

Lemma. *Let f and g be two embeddings of subspaces of dimensions k_S and k_Q ($k_Q \neq k_S$) in R^d into $R^{d'}$, i.e., $f : G(d, k_S) \rightarrow R^{d'}$ and $g : G(d, k_Q) \rightarrow R^{d'}$, where $G(d, k)$ represents the set of all k-dimensional subspaces in R^d. Suppose further that $f(S) = g(Q)$ whenever $Q \subset S$ or $S \subset Q$. Then, f and g must be trivial embeddings, i.e., $f(S) = g(Q)$ for all $S \in G(d, k_S)$ and $Q \in G(d, k_Q)$.*

Proof. Without loss of generality, assume that $k_Q < k_S$. Let S and S' be two subspaces in $G(d, k_S)$, then there exists a chain of subspaces $S = S_0, S_1, \ldots, S_n = S'$ ($n \leq k_S$) such that $dim(S_{i-1} \cap S_i) \geq k_Q$ and, consequently, there exists a corresponding chain $Q_1, \ldots, Q_n$ such that $Q_i \subseteq S_{i-1} \cap S_i$. Consequently, $f(S_0) = g(Q_1) = f(S_1) = g(Q_2) = \cdots = g(Q_n) = f(S_n)$. □

Corollary. *Let $u = f(S)$ and $v = g(Q)$ be nontrivial embeddings such that $\|u - v\|^2 = \mu \mathrm{dist}^2(S, Q) + \omega$ for all subspaces $S \in G(d, k_S)$ and $Q \in G(d, k_Q)$. Then, $\omega \neq 0$.*

Proof. Since $\mathrm{dist}(S, Q) = 0$ whenever either $Q \subset S$ or $S \subset Q$, then $\omega = 0$ would imply a trivial mapping. □

As a graphic illustration of the proof, consider the following example (Fig. 1): Let l be a subspace in the intersection of two higher dimensional subspaces π_1 and π_2 (i.e., $l \subset \pi_1$, $l \subset \pi_2$). Let $f(l)$ and $g(\pi_i), i = 1, 2$ be mappings such that $\omega = 0$. Since $l \subset \pi_1$, then $\mathrm{dist}(l, \pi_1) = 0$ and, hence, $f(l) = g(\pi_1)$. In addition, since $l \subset \pi_2$, then $\mathrm{dist}(l, \pi_2) = 0$ and, hence, $f(l) = g(\pi_2)$. We therefore immediately get that $g(\pi_1) = f(l) = g(\pi_2)$, i.e., all subspaces are mapped to the same point. This implies that a nontrivial mapping requires $\omega \neq 0$.

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 in the following section.

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 (5) lie on the intersection of a sphere and a hyperplane; they lie on a sphere since all share the same length $\|\mathbf{u}\|^2 = \frac{1}{2}k_S$, and 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. 2). 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. 2). 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 $\beta = -k_Q/(d\sqrt{2})$.

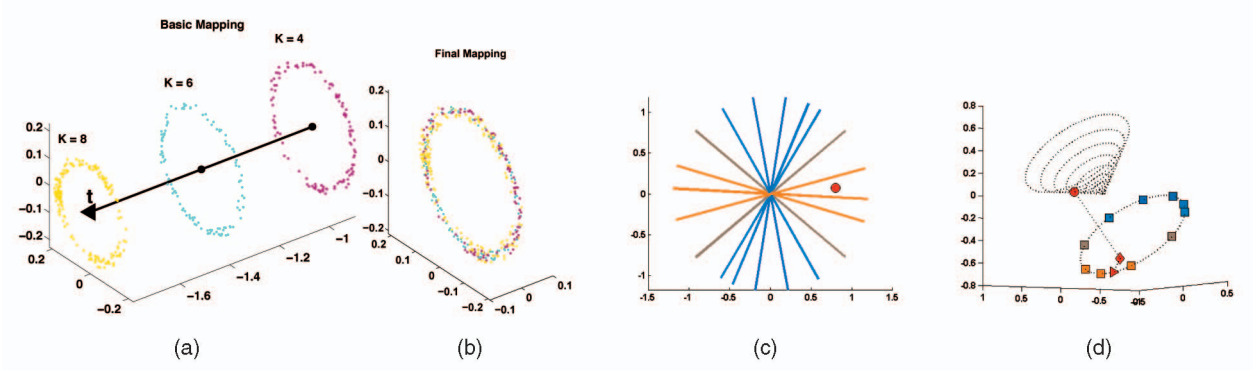


Fig. 2. The geometry of the mapped subspaces. (a) Slicing through mapped subspaces of intrinsic dimensions 4, 6, and 8 in a 10-dimensional space shows that the basic mapping of (5) maps subspaces of different dimensions onto different intersections of spheres and hyperplanes. (b) The refined mapping of (6) aligns these spheres. (c) Example of 1D subspaces in $\mathcal{R}^2$, color coded according to distance from a query point, and (d) shows their mapping. In the basic construction (5), the database lines and potential queries are mapped, respectively, to a ring and a cone in $\mathcal{R}^3$. The figure shows the query mapped to the cone, then projected to the hyperplane and scaled to lie on the ring.

Next, we wish to uniformly scale the query to bring it to the same sphere as the database items. Such uniform scaling also 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 $\|\mathbf{v} - \mathbf{u}\|_2^2 = \mu \operatorname{dist}^2(Q, S) + \omega$, 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 (6) reduces to the mapping of (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 The Case of Point Queries

The case of point queries is the simplest and most common in practice and, hence, received detailed analysis in [5]. In Section 3.1, we proposed an ostensibly different mapping which was shown to apply to both subspace queries as well as point queries. In fact, for the case of point queries, the mapping of (5) and that of [5] are equivalent, albeit in [5], it is formulated in terms of the null space of $\mathcal{S}$, while (5) is formulated in terms of the orthogonal basis spanning $\mathcal{S}$.

Thus, similarly to Basri et al. [5] and Section 3.3, we improve the mapping of (5) by applying the same procedure of translation and scaling. The resulting formula differs slightly since for points we have $\boldsymbol{t}^T\mathbf{v} = \|q\|^2/\sqrt{2}$, and therefore,

$$\mathbf{v} \doteq g(\boldsymbol{q}) = \frac{1}{c_q}\left(h(\boldsymbol{q}\boldsymbol{q}^T) - \frac{\|\boldsymbol{q}\|^2}{d\sqrt{2}}\boldsymbol{t}\right), \tag{7}$$

with $c_q = \sqrt{\frac{\|q\|^4}{2}(1 - \frac{1}{d})}$.

This mapping too implies $\|\mathbf{v} - \mathbf{u}\|_2^2 = \mu \operatorname{dist}^2(Q, S) + \omega$, where now

$$\mu = \frac{2d}{\|q\|^2\sqrt{k_S(d - k_S)(d - 1)}}$$

and $\omega = 2(1 - \sqrt{\frac{d - k_S}{k_S(d-1)}})$. In particular, $\mu = \frac{2d}{\|q\|^2(d-1)}$ and $\omega = 0$ when $k_S = 1$ for all d and $\mu \approx \frac{2}{\|q\|^2\sqrt{k_S}}$ and $\omega \approx 2(1 - \frac{1}{\sqrt{k_S}})$ when $k_S \ll d$.

4 Extensions to Additional Scenarios

The derivations and observations presented in the previous section apply to the case of a database of linear subspaces of a fixed intrinsic dimension. In this section, we build on these and propose mappings of the form presented in (1), extending them to three additional database types: subspaces of varying dimension, affine subspaces, and a database of points (where the queries maybe subspaces).

4.1 A Database of Subspaces of Varying Dimension

As previously mentioned, 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 recognition when varying numbers of images (from one to many) are available for different faces. The mapping of (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.

4.1.1 Query Dimension Larger than the Dimensions of Database Items

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.

$$\begin{aligned} \mathbf{u} &\doteq f(\mathcal{S}) = h(SS^T), \\ \mathbf{v} &\doteq g(\mathcal{Q}) = \frac{1}{2}h(QQ^T), \end{aligned} \tag{8}$$

and consequently, $\|\mathbf{v} - \mathbf{u}\|_2^2 = \frac{1}{8}k_Q + \frac{1}{2}k_S - \frac{1}{2}\|\cos\boldsymbol{\theta}\|^2 = \frac{1}{8}k_Q + \frac{1}{2}\operatorname{dist}^2(\mathcal{Q}, \mathcal{S})$. This implies we have obtained a mapping for which $\|\mathbf{v} - \mathbf{u}\|_2^2 = \mu \operatorname{dist}^2(\mathcal{Q}, \mathcal{S}) + \omega$, 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 (8) can be used for all subspaces even when their intrinsic dimensions vary.

4.1.2 Query Dimension Smaller than Database Items Dimension

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:

$$\begin{aligned} \mathbf{u} &\doteq f(\mathcal{S}) = [h(SS^T), \;\; \sqrt{0.5(k_{\max} - k_S)}], \\ \mathbf{v} &\doteq g(\mathcal{Q}) = [h(QQ^T), \;\; 0], \end{aligned} \tag{9}$$

where $k_{\max} = \max_S k_S$. Consequently, $\|\mathbf{v} - \mathbf{u}\|_2^2 = \mu \operatorname{dist}^2(\mathcal{Q}, \mathcal{S}) + \omega$, where $\mu = 1$ and $\omega = \frac{1}{2}k_{\max} - \frac{1}{2}k_Q$. Hence, (9) 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.

4.1.3 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 preprocessing the database twice, once with the mapping of (8), which is appropriate for database subspaces with $k_Q > k_S$, and once with the mapping of (9), 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 preprocessing time and memory requirement, but it does make the running time slightly slower. Still, it is much faster than full linear search (see Section 5).

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 preprocessing 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 preprocess 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 preprocessed databases. Note that at most we need to preprocess the database for $\max(k_S) - \min(k_S) + 1$ values of k_Q. This increases the preprocessing 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.

Finally, we note that refining the mapping, as was proposed in (6), is impossible in this case since the database subspaces are mapped onto different hyperplanes, depending on k_S.

4.2 A Database of Affine Subspaces

The case of affine subspace differs from that of linear subspaces. A popular measure is the minimal euclidean distance (i.e., the minimum distance between any pair of points of the two subspaces). This measure, however, ignores the angular difference between spaces and therefore is irrelevant for certain applications. Furthermore, since euclidean distances between affine subspaces do not form a metric (no triangle inequality), one cannot map affine subspaces into a metric space of points (which we use). An alternative distance considers only the angular difference. This reduces the problem to the case of linear subspaces that was addressed in the previous sections. Finally, one could propose measures which combine the euclidean and angular distances; however, it is not clear what the practical uses of such measures are. Therefore, we separate between two cases, that of point queries, where the euclidean distance is appropriate, and the case of affine subspace queries, where we propose a distance measure appropriate for a particular application.

4.2.1 Point Queries

An affine subspace $\mathcal{A}$ that is parallel to a linear subspace $\mathcal{S}$, given by a $d \times k$ matrix S with orthonormal columns, can be represented as an intersection of affine hyperplanes, parameterized by a $d \times (d-k)$ matrix Z with orthonormal columns such that $ZZ^T = I_d - SS^T$ (I_d representing the $d \times d$ identity matrix), and a vector of offset values $\mathbf{t} \in \mathcal{R}^{d-k}$. Below, we denote by $\hat{Z}$ the $(d+1) \times (d-k)$ matrix whose first d rows contain Z and last row contains $\mathbf{t}^T$, i.e., $\hat{Z} = \left[\begin{smallmatrix} Z \\ \mathbf{t}^T \end{smallmatrix}\right]$. Given a query $\mathbf{q} \in \mathcal{R}^d$ we use homogenous coordinates, denoting $\hat{\mathbf{q}} = (\mathbf{q}^T, 1)^T$.

We use $\hat{Z}$ and $\hat{\mathbf{q}}$ to define a mapping similar to that of linear spaces. The columns of $\hat{Z}$ are not orthonormal due to the additional last row. We can account for this by introducing an additional entry, as follows:

$$\begin{aligned} \mathbf{u} &= \hat{f}(\mathcal{A}) = -(h(\hat{Z}\hat{Z}^T), \hat{c}(\mathcal{A})) \in \mathcal{R}^{\hat{d}'}, \\ \mathbf{v} &= \hat{g}(\mathbf{q}) = (h(\hat{\mathbf{q}}\hat{\mathbf{q}}^T), 0) \in \mathcal{R}^{\hat{d}'}. \end{aligned} \tag{10}$$

$\mathbf{u}$ and $\mathbf{v}$ lie in $\mathcal{R}^{\hat{d}'}$, where now $\hat{d}' = (d+1)(d+2)/2 + 1$. The last entry is added to make the norm of $\mathbf{u}$ equal across the database. To achieve this, we set

$$\hat{c}(\mathcal{A}) = \sqrt{(M^4 - \|\hat{Z}\hat{Z}^T\|_F^2)/2},$$

where $\|\hat{Z}\hat{Z}^T\|_F^2 = \|ZZ^T\|_F^2 + 2\|Z\mathbf{t}\|^2 + \|\mathbf{t}\|^2 = d - k + 3\|\mathbf{t}\|^2$ and M is a positive constant; M must be sufficiently large to allow taking the square root for all the affine subspaces in the database (thus it is determined by the affine space with largest $\|\mathbf{t}\|$). Note that we set the last entry of $\mathbf{v}$ to zero so that the last entry of $\mathbf{u}$ does not affect the inner product of $\mathbf{u}^T\mathbf{v}$. Therefore, $\|\mathbf{u}\|^2 = (1/2)M^4$, $\|\mathbf{v}\|^2 = (1/2)\|\hat{\mathbf{q}}\|^4$, and $\mathbf{u}^T\mathbf{v} = -\frac{1}{2}\hat{\mathbf{q}}^T\hat{Z}\hat{Z}^T\hat{\mathbf{q}} = -\frac{1}{2}\mathrm{dist}^2(\mathbf{q}, \mathcal{A})$, where $\mathrm{dist}(\mathbf{q}, \mathcal{A})$ denotes the euclidean distance between $\mathbf{q}$ and $\mathcal{A}$, and consequently, we obtain

$$\|\mathbf{u} - \mathbf{v}\|^2 = \mathrm{dist}^2(\mathbf{q}, \mathcal{A}) + \frac{1}{2}(M^4 + \|\hat{\mathbf{q}}\|^4), \tag{11}$$

where the additional constant depends on the query point $\mathbf{q}$ and is independent of the database subspace $\mathcal{A}$.

Similarly to the case of linear subspaces, the affine subspaces too are mapped to the intersection of a sphere (of radius $M^2/\sqrt{2}$) and a hyperplane $-\sqrt{2}\, \boldsymbol{t}^T\mathbf{u} = d - k$, and so the query can be projected into this hyperplane in the same way as in Section 3.3.

4.2.2 Affine Subspace Queries

As previously mentioned, we know of no accepted distance measure between two affine subspaces. This is probably because affine subspaces are defined by two different components with different natures: a linear subspace 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 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 [8]. Having reduced the problem to that of linear subspaces, we can use the appropriate mapping as proposed in the previous sections.

4.3 A Database of Points

Our approach allows us to also handle cases where the roles of database and queries are reversed; the database now containing points and the queries are the subspaces. Examples for such cases include systems which compare noisy incoming signals to "clean" examples. Indeed, such scenarios were the motivation behind the study of the "partial match" problem on strings [15].

To handle point database elements, we apply the following heuristic. We add a component to each database point, thus increasing the ambient space to dimension $d+1$. Let p be a database point and S be a linear query subspace. We replace $\mathbf{p}$ by $\hat{\mathbf{p}} = (\mathbf{p}^T, a)^T$, where

$$a = \sqrt{-\|\mathbf{p}\|^2 + \sqrt{2\|\mathbf{p}\|^2 + c}}, \tag{12}$$

subject to $c \geq \max_{\mathbf{p}} \|\mathbf{p}\|^4 - 2\|\mathbf{p}\|^2$. For S, we define $\hat{S}$ by adding a zero row at its bottom.

Consequent to this construction, we get

$$\|\hat{\mathbf{p}}\hat{\mathbf{p}}^t - \hat{S}\hat{S}^t\|^2 = c + k_s + 2\mathrm{dist}^2(\mathbf{p}, S), \tag{13}$$

and we can further eliminate redundancies by using the mappings $\mathbf{u} = f(\hat{\mathbf{p}}) = h(\hat{\mathbf{p}}\hat{\mathbf{p}}^T)$ and $\mathbf{v} = g(\hat{\mathcal{S}}) = h(\hat{\mathcal{S}}\hat{\mathcal{S}}^T)$. Note that if all $\|\mathbf{p}\| \leq \sqrt{2}$, then we can use $c = 0$ and, consequently,

$$a = \sqrt{-\|\mathbf{p}\|^2 + \sqrt{2}\|\mathbf{p}\|}.$$

A similar construction can be developed for the case of an affine subspace query and a database of points. For a database point $\mathbf{p}$, we add two new entries (increasing the ambient dimension to $d+2$) as follows: $\tilde{\mathbf{p}} = (\mathbf{p}^T, 1, \tilde{a})^T$, where $\tilde{a}^2 = \tilde{c} - \|\mathbf{p}\|^2 - 1$ with a sufficiently large constant c. The affine subspace query is represented by a $(d+2) \times (d-k)$ matrix $\tilde{Z}$ composed of the matrix $\hat{Z}$ (defined in the previous section) to which we add a row of zeros at its bottom. Consequently,

$$\|\tilde{\mathbf{p}}\tilde{\mathbf{p}}^t + \tilde{Z}\tilde{Z}^t\|^2 = 2\mathrm{dist}^2(\mathbf{p}, S) + c^2 + d - k_s + 3\|\mathbf{t}\|^2, \tag{14}$$

and the constants are mutual for all the points in the database. As in the linear case, we can further eliminate redundancies by using the mappings $\mathbf{u} = f(\tilde{\mathbf{p}}) = h(\tilde{\mathbf{p}}\tilde{\mathbf{p}}^T)$ and $\mathbf{v} = g(\tilde{Z}) = -h(\tilde{Z}\tilde{Z}^T)$.

5 Complexities

Given a query, our search routine starts by mapping it to $O(d^2)$-space using (5) or (6) (or (8), (9) 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., [3], [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 $\mathcal{R}^{d^2}$. Note that here we take $k_Q = 1$ for point queries.

One ANN method is the search-tree-based approach (e.g., [3]). 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. The runtime of [3] 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 an $O(\log n/\epsilon)$ 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 [3], 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 [3], used in our experiments.

Note that an exact sequential search for a nearest subspace using the distance $\|QQ^T - SS^T\|_F^2$ requires $O(k_Q d^2 + nk_S d^2)$. 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, e.g., Fig. 3 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 $O(n(k_Q k_S d + \max(k_Q, k_S)^3))$ and is therefore preferable when both $k_Q, k_S \ll d$.

Our formulation maps subspaces and points of dimension d to points of dimension $d' = O(d^2)$. As was shown by

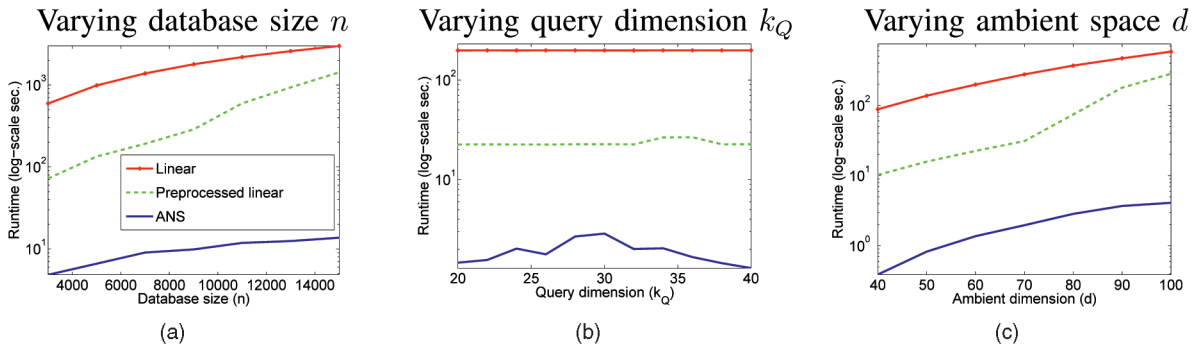


Fig. 3. Synthetic data tests. Log-scale runtimes compared for an exact linear search, linear search with a preprocessed database, and our ANS method with ANN $\epsilon = 100$. The following tests were performed: (a) Varying n: Database subspaces of $k_S = 30$ embedded in $d = 60$ space, tested with 1,000 queries of dimension $k_Q = 10$. (b) Varying k_Q: Database contains 1,000 subspaces with $k_S = 30$, tested with 1,000 queries. (c) Varying d: Database contains 1,000 subspaces with $k_S = 30$, tested with 1,000 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.

Muja and Lowe [27], this higher dimensionality does not pose a problem when applying kd-tree-based approximate nearest neighbor search. Nevertheless, in many vision applications, this maybe intolerably large. We can approach this problem by projecting the mapped database and query onto a space of lower dimension and applying nearest neighbor search to the projected points. Random projections are commonly used in ANN searches whenever $d \gg \log n$. For a set of n points, the celebrated Johnson-Lindenstrauss Lemma [24] guarantees with high probability that a random projection into $O(\epsilon^{-2} \log n)$ dimension does not distort distances by more than a factor of $1 + \epsilon$. Magen [26] (see also [28]) has extended this lemma to affine spaces, showing that, for a set of n affine spaces of rank k, a random projection into $O(\epsilon^{-3} k \log(kn))$ dimension distorts distances by no more than a factor of $1 + \epsilon$. Utilizing these results, we can first map the subspaces in the database to points and then project to a lower dimensional space which is logarithmic in n. Alternatively, we can first project the subspaces to a space of lower dimension and then map the projected subspaces to points, this time obtaining a polylogarithmic dimension in n.

In the experiments reported below, we have sometimes found that good results can be obtained with significant speedup, if both the database and queries are first projected to a low dimension, before mapping. We do this for N_P projections, each of dimension b. We perform multiple projections as any particular projection may distort the distances between the query and database objects. Thus, on each random projection, we extract c approximate nearest neighbors and compute the true distance between the query and all cN_P candidates. Finally, we report the closest match across all these projections. Our overall query running time is thus $N_P(O(bdk_Q) + O(k_Q b^2) + T_{ANN}(n, b^2) + cO(d^2 k))$, where $O(bdk_Q)$ is the time for projecting onto a b dimensional subspace, and $O(dk)$ the time for measuring the true distance between the query and a candidate database subspace ($k = min(k_Q, k_S)$). This is summarized in Table 2.

6 Experiments

To evaluate the performance of our ANS scheme, we adopt the conventional tests in the field (e.g., [1], [3]). These are based on synthetic data with varying parameters as this is the best 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 near subspace, with only a small penalty in accuracy. Our implementation is in C and uses the ANN kd-tree code of [3]. OpenCV was used for all of our matrix routines.

6.1 Synthetic Data

6.1.1 Subspace Queries

We tested our ANS scheme on data sets containing thousands of synthetically produced queries and database elements, of fairly large dimensions (Fig. 3). The tests compare our ANS scheme to a linear search and a linear search with a preprocessed database (see Section 5). Runtimes 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 [3], [25], 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 ofline. Our tests demonstrate that the ANS scheme is significantly faster than both linear search methods. In addition, in all of 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

TABLE 2
Summary of Complexities

Method	Query time	Pre-processing time	Space
Our	$O(k_Q d^2) + T_{ANN}(n, d^2)$	$O(nk_S d^2) + T_{pANN}(n, d^2)$	$O(nd^2)$
Exact	$O(k_Q d^2) + O(d^2 n)$	$O(nk_S d^2)$	$O(nd^2)$

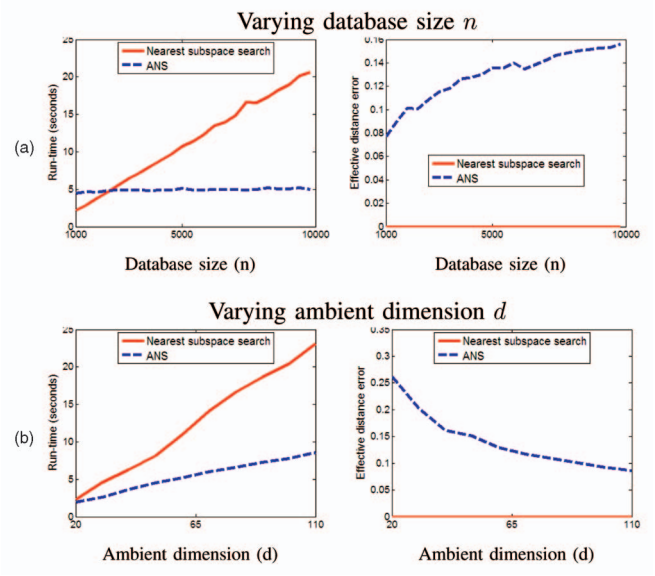


Fig. 4. Synthetic data, *point* queries with linear subspace data sets. (a) Comparing ANS with nearest subspace search, where ambient dimension is $d = 60$ and database subspace dimension is $k_S = 4$. (b) Comparing ANS with nearest subspace search, where database size is $n = 5{,}000$ and database subspace dimension is $k_S = 4$.

considering the high value for ϵ is a well documented property of the kd-trees method [3]. This is a point worth noting, especially since statistics of our data show that, on average, each database item has approximately 0.3 percent neighbors within $1 + 0.01$ times the distance to the nearest neighbor.

6.1.2 Point Queries

Similar tests were applied also for the case of point queries. Here, random projections were used to speed up query runtimes. We used $N_P = 23$ random projections, measuring the true distance to the best $c = 15$ subspaces in each projection and reporting the best one. Database subspaces were selected uniformly, at random. Following [38], we generate point queries such that at least one database subspace is at a distance of no more than $(1 + \eta)2R\sqrt{d}$ from each query, where $R = 0.1$ and $\eta = 0.0001$. The results, presented in Fig. 4, show that our algorithm is faster than sequential database scan, while maintaining fairly low Err rates.

6.2 Image Approximation

We next demonstrate how subspaces can be used to capture local translations of intensity patches. Our goal here is to approximate the intensities of a query image by tiling it with intensity patches obtained from an image of an altogether different scene. A similar procedure is frequently used in the so-called "by-example" patch-based methods for applications including segmentation [12] and reconstruction [21].

A collection of 1,000 points were selected at random in a single image (Fig. 5). Then, 16 different, overlapping 5×5 patches around each coordinate were used to produce a $k = 4$ subspace by taking their four principal components. These were stored in our *subspace* database. In addition, all 16 patches were stored for our *point* (patch) database.

Given a novel test image, we subdivided it into a grid of nonoverlapping 5×5 patches. For each such patch, we searched the point database for a similar patch using sequential (exact) and point-ANN-based search methods. The selected database patch was then used as an approximation to the original input patch. Similarly, we used both sequential (exact) and ANS search schemes for selecting a matching subspace in the subspace database for each patch. The point on the selected subspace, closest to the query patch, was then taken as its approximation.

Fig. 6 presents the results obtained by each of the methods.[1] Note the improved quality of the subspace-based reconstructions over the point-based methods, evident also in the mean L1 error reported in Fig. 5. In addition, with the exception of the point ANN method, which did the worst in terms of quality, our ANS method was fastest, implying that an ANS method can be used to quickly and accurately capture local translations of image patches.

6.3 Scene Classification

We next test our method on real image data using the scene classification data of [18]. We randomly selected 10 "training" images and 10 (different) "testing" images of three categories. Fifty random coordinates were selected in each image. Then, nine different, overlapping 9×9 patches around each coordinate were used to produce a $k = 5$ subspace by taking their five 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. Fig. 7 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 of 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.

6.4 Speaker Recognition

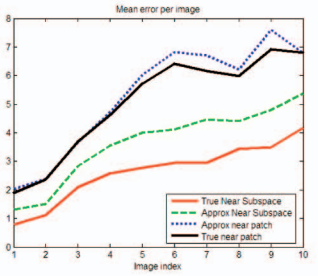
We tested our method on voice data from [11], consisting of 31 subjects uttering the same short phrase (2-3 seconds long), three times over a phone connection (a total of 93 samples). Each sample was represented by standard mel-frequency cepstrum frame descriptors for time frames of 25 msec, with overlaps of 50 percent. 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 three consecutive time frames were taken as points. Twenty such points, each point overlapping its neighbors by two time frames, were used to produce a single linear subspace of dimension $k_Q = k_S = 13$. Each sample thus contributed

1. Additional image approximation results can be downloaded from http://www.wisdom.weizmann.ac.il/~vision/ANS/.

(a)

Method	*Run time*
Approx nearest patch	0.6 sec.
Approx nearest subspace	1.2 sec.
(Exact) Nearest subspace	4.2 sec.
(Exact) Nearest patch	27.7 sec.

(b)



(c)

Fig. 5. Image reconstruction details. (a) The single database image used for reconstruction, (b) mean runtimes for each method, and (c) L1 error reconstruction error for each of the test images. Both database and query images were taken from the Corel data set.

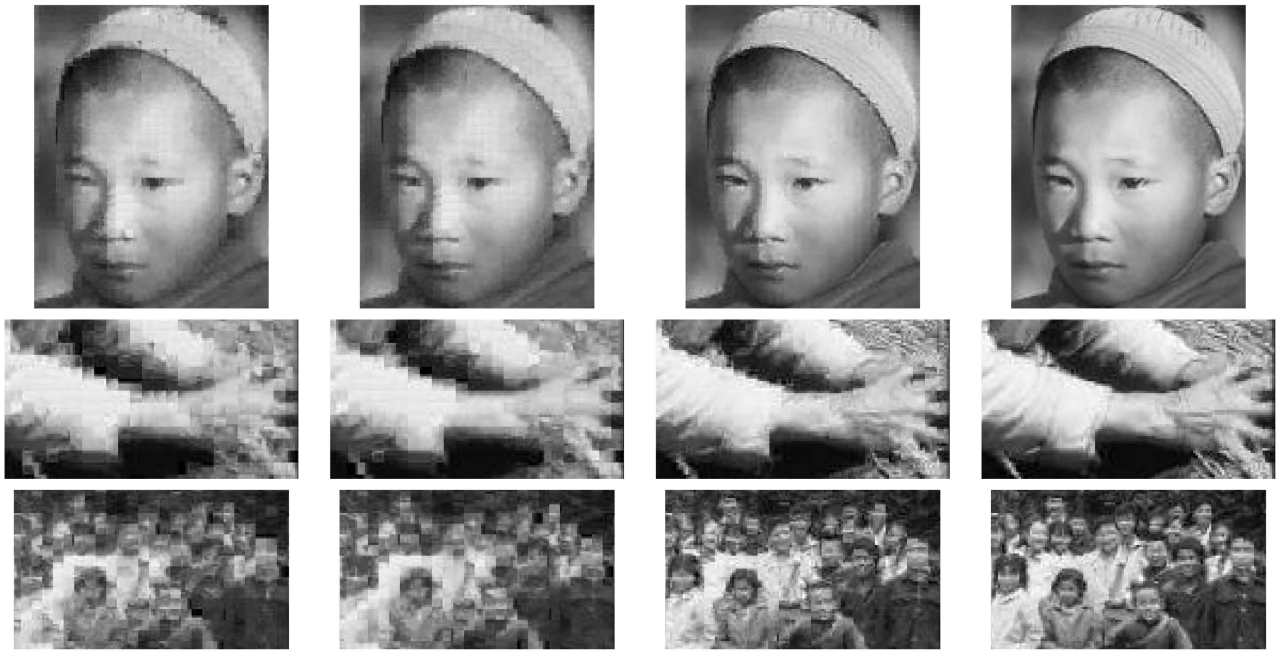


Fig. 6. Image reconstruction results. Reconstructed using a single outdoor scene image. See Fig. 5 for runtimes and error rates.

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. 7. Numerical results on real data. ANS is approximately an order of magnitude faster with no loss of accuracy.

20-30 such subspaces for a total of 1,280 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 percent; however, even on such a small database, our ANS search was an order of magnitude faster (see Fig. 7). 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 [11].

6.5 Yale-B Face Recognition

Subspaces are commonly used to capture the appearance of faces under varying illuminations in recognition systems (e.g., [7], [20]). Here, we test the performance of our approach on a similar application using real data from the Yale-B face data set [20] (see Fig. 9 for example images).

For every subject and pose combination in the Yale-B data set, we randomly chose 18 illuminations as the database

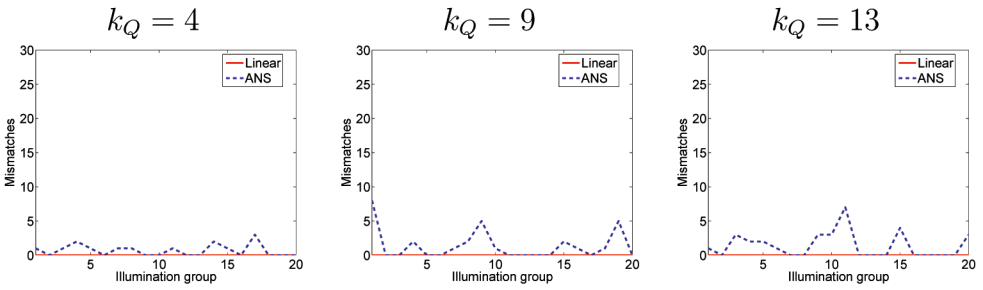


Fig. 8. Yale-B face recognition. Subject mismatches on the Yale-B database [20]. A database of 90 subspaces was produced by fitting subspaces with $k_S = 9$ to 18 randomly selected illuminations for each subject+pose combination. Ninety queries were likewise produced by fitting subspaces of $k_Q =$ 4, 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 rates for our method is 99.2 percent, 98.4 percent, and 98.4 percent, for $k_Q =$ 4, 9, and 13, respectively. Linear search resulted in zero errors; hence, the corresponding line unites with the x-axis in the figures.

Fig. 9. Yale-B faces. Example images from the Yale-B data set [20].

examples, and fit a subspace of dimension $k_S = 9$ to these images. Since the information available at query time may vary, we ran tests with query subspaces of dimensions 4, 9, and 13. For each test, we randomly select sets of 4, 9, or 13 images not used for the database, from each subject+pose combination, and use them to fit the query subspaces. Our goal is to recognize the correct face under these conditions. Fig. 8 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 percent, 98.4 percent, and 98.4 percent, 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 of our tests, we used an ANN $\epsilon = 10$. Better results can be obtained, at the price of slower processing speeds, for lower ϵ values.

6.6 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 [31]. 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 3D subspace.

In our tests, we use the action database of [9] (see example frames in Fig. 10), containing 10 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 $\mathbf{G_P}$ of dimension 147×3, which we then orthonormalize using SVD, for a total of 3,200 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 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 percent (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 suitable selection of ANN parameters. Note that although [9] report better classification results of 97 percent, their method takes advantage of additional information. In particular, unlike their method, ours was applied to the unsegmented raw data.

Fig. 10. Motion-based action recognition. Sample frames (cropped) from the action database of [9] used in our action recognition tests.

7 Conclusions

We have presented a sublinear, 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 is useful for a wide range of applications, and indeed demonstrated its capabilities in a range of experiments.

Although already a practical solution, challenges still lie ahead. The reduction to points introduces an error which deteriorates the quality of results. We therefore wish to further search for better reductions with lower errors. It would also be interesting to design solutions tailored for the case of subspaces, and not via a reduction to points. Finally, not only points and subspaces are used to represent information in pattern recognition applications; other frequently used geometric entities include, for example, multidimensional Gaussians and spheres. We are therefore currently interested in looking for efficient techniques for searching through databases of objects of such kinds.

Acknowledgments

The research of Lihi Zelnik-Manor is supported by Marie Curie IRG-208529. The vision group at the Weizmann Institute is supported in part by the Moross Foundation. This research was performed partly while Ronen Basri was at the Toyota Technological Institute at Chicago. This manuscript contains results that have previously appeared in the *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*) (2007) [5] and the *Proceedings of the International Workshop on Subspace Methods at the IEEE International Conference on Computer Vision* (*ICCV*) (2009) [6]. Author names in alphabetical order due to equal contribution.

References

[1] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *Proc. 47th Ann. IEEE Symp. Foundations of Computer Science,* pp. 459-468, 2006.

[2] A. Andoni, P. Indyk, R. Krauthgamer, and H.L. Nguyen, "Approximate Line Nearest Neighbor in High Dimensions," *Proc. 20th Ann. ACM-SIAM Symp. Discrete Algorithms,* pp. 293-301, 2009.

[3] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions," *J. ACM,* vol. 45, no. 6, pp. 891-923, www.cs.umd.edu/mount/ANN/, 1998.

[4] J.J. Atick, P.A. Griffin, and A.N. Redlich, "Statistical Approach to Shape from Shading: Reconstruction of Three-Dimensional Face Surfaces from Single Two-Dimensional Images," *Neural Computation,* vol. 8, no. 6, pp. 1321-1340, 1996.

[5] R. Basri, T. Hassner, and L. Zelnik-Manor, "Approximate Nearest Subspace Search with Applications to Pattern Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.

[6] R. Basri, T. Hassner, and L. Zelnik-Manor, "A General Framework for Approximate Nearest Subspace Search," *Proc. IEEE Int'l Workshop Subspace Methods at IEEE Int'l Conf. Computer Vision,* Sept. 2009.

[7] R. Basri and D. Jacobs, "Lambertian Reflectance and Linear Subspaces," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 2, pp. 218-233, Feb. 2003.

[8] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, "Performance of Optical Flow Techniques" *Int'l J. Computer Vision,* vol. 12, no. 1, pp. 43-77, 1994.

[9] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as Space-Time Shapes," *Proc. 10th IEEE Int'l Conf. Computer Vision,* www.cs.weizmann.ac.il/~vision/SpaceTimeActions.html, pp. 1395-1402, 2005.

[10] V. Blanz and T. Vetter, "Face Recognition Based on Fitting a 3D Morphable Model," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 9, pp. 1063-1074, Sept. 2003.

[11] O. Boiman and M. Irani, "Similarity by Composition," *Proc. Neural Information Processing Systems,* vol. 19, pp. 177-184, 2006.

[12] E. Borenstein and S. Ullman, "Learning to Segment," *Proc. Eighth European Conf. Computer Vision,* vol. 3023, pp. 315-328, 2004.

[13] M.E. Brand, "Morphable 3D Models from Video," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 456-463, 2001.

[14] C. Bregler, A. Hertzmann, and H. Biermann, "Recovering Non-Rigid 3D Shape from Image Streams," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 690-696, 2000.

[15] M. Charikar, P. Indyk, and R. Panigrahy, "New Algorithms for Subset Query, Partial Match, Orthogonal Range Searching, and Related Problems," *Proc. 29th Int'l Colloquium on Automata, Languages and Programming,* pp. 451-462, 2002.

[16] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-Sensitive Hashing Scheme Based on P-Stable Distributions," *Proc. 20th Ann. Symp. Computational Geometry,* pp. 253-262, 2004.

[17] A. Edelman, T.A. Arias, and S.T. Smith, "The Geometry of Algorithms with Orthogonality Constraints," *SIAM J. Matrix Analysis and Applications,* vol. 20, no. 2 pp. 303-353, 1999.

[18] L. Fei-Fei and P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[19] A. Fitzgibbon and A. Zisserman, "Joint Manifold Distance: A New Approach to Appearance Based Clustering," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 26-33, 2003.

[20] 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 Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 6, pp. 643-660, June 2001.

[21] T. Hassner and R. Basri, "Example Based 3D Reconstruction from Single 2D Images," *Proc. Beyond Patches Workshop, IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[22] P. Indyk, R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," *Proc. 30th Ann. ACM Symp. Theory of Computing,* pp. 604-613, 1998.

[23] M. Irani and P. Anandan, "Factorization with Uncertainty," *Proc. Sixth European Conf. Computer Vision,* vol. 1, pp. 539-553, 2000.

[24] W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz Maps into a Hilbert Space," *Contemporary Math,* pp. 189-206, vol. 26, 1984.

[25] T. Liu, A.W. Moore, A. Gray, and K. Yang, "An Investigation of Practical Approximate Nearest Neighbor Algorithms," *Proc. Neural Information Processing Systems,* pp. 825-832, 2004.

[26] A. Magen, "Dimensionality Reductions That Preserve Volumes and Distance to Affine Spaces, and Their Algorithmic Applications," *Randomization and Approximation Techniques in Computer Science,* pp. 239-253, Springer, 2002.

[27] M. Muja and D.G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," *Proc. Int'l Conf. Computer Vision Theory and Applications,* pp. 331-340, 2009.

[28] P. Indyk and A. Naor, "Nearest Neighbor Preserving Embeddings," *ACM Trans. Algorithms,* vol. 3, no. 3, 2007.

[29] R. Ramamoorthi and P. Hanrahan, "On the Relationship between Radiance and Irradiance: Determining the Illumination from Images of Convex Lambertian Object," *J. Optical Soc. of Am.,* vol. 18, no. 10, pp. 2448-2459, 2001.

[30] *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice,* G. Shakhnarovich, T. Darrell, and P. Indyk, eds., MIT Press, 2006.

[31] 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 Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 11, pp. 2045-2056, Nov. 2007.

[32] 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,* pp. 227-239, Springer, 1998.

[33] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams under Orthography: A Factorization Method," *Int'l J. Computer Vision*, vol. 9, no. 2, pp. 137-154, 1992.
[34] L. Torresani, D. Yang, G. Alexander, and C. Bregler, "Tracking and Modeling Non-Rigid Objects with Rank Constraints," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 493-500, 2001.
[35] S. Ullman and R. Basri, "Recognition by Linear Combinations of Models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 992-1006, Oct. 1991.
[36] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210-227, Feb. 2009.
[37] O. Yamaguchi, K. Fukui, and K. Maeda, "Face Recognition Using Temporal Image Sequence," *Proc. Third Int'l Conf. Face and Gesture Recognition*, pp. a318-323, 1998.
[38] P.N. Yianilos, "Locally Lifting the Curse of Dimensionality for Nearest Neighbor Search," *Proc. 11th Ann. ACM-SIAM Symp. Discrete Algorithms*, extended abstract, pp. 361-370, 2000.
[39] H. Zhang, A.C. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 2126-2136, 2006.

Ronen Basri received the BSc degree in mathematics and computer science from Tel Aviv University in 1985 and the PhD degree from the Weizmann Institute of Science in 1991. From 1990 to 1992, he was a postdoctoral fellow at the Massachusetts Institute of Technology in the Department of Brain and Cognitive Science and the Artificial Intelligence Laboratory under the McDonnell-Pew and Rothchild programs. Since then, he has been affiliated with the Weizmann Institute of Science in the Department of Computer Science and Applied Mathematics, where he currently holds the position of professor and the incumbent of the Elaine and Bram Goldsmith Chair of Applied Mathematics. In 2007, he served as chair for the Department of Computer Science and Applied Mathematics. He further held visiting positions at the NEC Research Institute in Princeton, New Jersey, Toyota Technological Institute at Chicago, the University of Chicago, and the Janelia Farm Campus of the Howard Hughes Medical Institute. His research has focused on computer vision, especially in the areas of object recognition, shape reconstruction, lighting analysis, and image segmentation. His work deals with the development of algorithms, analysis, and implications to human vision. He is a member of the IEEE.

Tal Hassner received the BA degree in computer science from the Academic College of Tel-Aviv Yaffo in 1998, and the MSc and PhD degrees in applied mathematics and computer science from the Weizmann Institute of Science in 2002 and 2006, respectively. He later completed a postdoctoral fellowship, also at the Weizmann institute. In 2006, he joined the faculty at the Open University of Israel. In addition, he has been an adjunct faculty member at the Academic College of Tel-Aviv Yaffo since 2006. His distinctions include the best Student Paper Award at the IEEE Shape Modeling International Conference 2005 and the AIM@SHAPE Best Paper Award 2005. His research interests are in the areas of computer vision, including face recognition and single-view, 3D-reconstruction.

Lihi Zelnik-Manor received the BSc degree in mechanical engineering from the Technion in 1995, where she graduated summa cum laude, and the MSc (with honors) and PhD degrees in computer science from the Weizmann Institute of Science in 1998 and 2004, respectively. After graduating, she worked as a postdoctoral fellow in the Department of Engineering and Applied Science at the California Institute of Technology (Caltech). Since 2007, she has been a senior lecturer in the Electrical Engineering Department at the Technion. Her research focuses on the analysis of dynamic visual data, including video analysis and visualizations of multiview data. Her awards and honors include the Israeli high-education planning and budgeting committee (Vatat) three-year scholarship for outstanding PhD students, and the Sloan-Swartz postdoctoral fellowship. She also received the best Student Paper Award at the IEEE Shape Modeling International Conference 2005 and the AIM@SHAPE Best Paper Award 2005. She is a member of the IEEE.