Approximate Nearest Subspace Search with Applications to Pattern Recognition

Ronen Basri† Tal Hassner† Lihi Zelnik-Manor‡
† Weizmann Institute of Science ‡ California Institute of Technology
Rehovot, Israel Pasadena, CA 91125, USA
{ronen.basri, tal.hassner}@weizmann.ac.il lihi@vision.caltech.edu

Abstract

Linear and affine subspaces are commonly used to describe appearance of objects under different lighting, viewpoint, articulation, and identity. A natural problem arising from their use is – given a query image portion represented as a point in some high dimensional space – find a subspace near to the query. This paper presents an efficient solution to the approximate nearest subspace problem for both linear and affine subspaces. Our method is based on a simple reduction to the problem of nearest point search, and can thus employ tree based search or locality sensitive hashing to find a near subspace. Further speedup may be achieved by using random projections to lower the dimensionality of the problem. We provide theoretical proofs of correctness and error bounds of our construction and demonstrate its capabilities on synthetic and real data. Our experiments demonstrate that an approximate nearest subspace can be located significantly faster than the exact nearest subspace, while at the same time it can find better matches compared to a similar search on points, in the presence of variations due to viewpoint, lighting etc.

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 [4, 18], viewpoint [21, 19], articulation [7, 20], and even identity [3, 5]. Typically, given a query image portion, represented as a point 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?

R.B and T.H. were supported in part by the A.M.N. Fund for the promotion of science, culture and arts in Israel and by the European Community grant IST-2002-506766 Aim@Shape. The vision group at the Weizmann Institute is supported in part by the Moross Foundation. L.Z.-M. was supported by ONR grant N00014-06-1-0734 and NSF grant CMS-0428075. Author names are ordered alphabetically due to equal contribution.

The related problem of finding the nearest neighbor within a database of high dimensional points 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., [2, 8, 11, 13]). These algorithms achieve sub-linear search times when locating a near, not necessarily the nearest neighbor, suffices. 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 a query point and a database of n subspaces of dimension k embedded in $\mathcal{R}^d$, ANS query running time, using our construction, is $O(kd^2) + T_{ANN}(n, d^2)$, 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 achieve further speedup by using random projections to lower the dimensionality of the problem. Our method is related to recent work by Magen [15], who reduced ANS to a nearest hyperplane search. Magen's method, however, requires $O(n^{d^2})$ preprocessing time and space while our preprocessing requires only $O(nkd^2)$.

We next describe our method, provide theoretical proofs of correctness and error bounds of our construction, and present both analytical and empirical analysis. We further demonstrate our method's capabilities on synthetic and real data, with an emphasis on image and image-patch retrieval applications.

2. Nearest Subspace Search

The *nearest subspace problem* is defined as follows. Let $\{\mathcal{S}^1, \mathcal{S}^2, ..., \mathcal{S}^n\}$ be a collection of *linear* (or *affine*) subspaces in $\mathcal{R}^d$, each with intrinsic dimension k. Then, given a query point $\mathbf{q} \in \mathcal{R}^d$, denote by $\mathrm{dist}(\mathbf{q}, \mathcal{S}^i)$ the Euclidean distance between the query $\mathbf{q}$ and a subspace $\mathcal{S}^i, 1 \le i \le n$, we seek the subspace $\mathcal{S}^*$ that is nearest to $\mathbf{q}$, i.e., $\mathcal{S}^* = \arg\min_i \mathrm{dist}(\mathbf{q}, \mathcal{S}^i)$.

1-4244-1180-7/07/$25.00 ©2007 IEEE

We approach the nearest subspace problem by reducing it to the well explored nearest neighbor (NN) problem for points. To that end we seek to define two transformations, $\mathbf{u} = f(\mathcal{S})$ and $\mathbf{v} = g(\mathbf{q})$, which respectively map any given subspace $\mathcal{S}$ and query point $\mathbf{q}$ to points $\mathbf{u}, \mathbf{v} \in \mathcal{R}^{d'}$ for some d', such that the distance $\|\mathbf{v} - \mathbf{u}\|$ increases monotonically with $\mathrm{dist}(\mathbf{q}, \mathcal{S})$. In particular, we derive below such transformations for which $\|\mathbf{v} - \mathbf{u}\|^2 = \mu\,\mathrm{dist}^2(\mathbf{q}, \mathcal{S}) + \nu$ for some constants μ and ν. This is summarized below.

Linear Subspaces: We represent a linear subspace $\mathcal{S}$ by a $d \times k$ matrix S with orthonormal columns. Our transformations map $\mathcal{S}$ and $\mathbf{q}$ onto $\mathcal{R}^{d'}$ with $d' = d(d+1)/2$. 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.,

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

Finally, we denote by $\mathbf{n} = \sqrt{2}h(I) \in \mathcal{R}^{d'}$ (I denotes the identity matrix) a vector whose entries are one for each diagonal entry in $h(.)$ and zero elsewhere. We are now ready to define our mapping. Denote by

$$\begin{aligned} \mathbf{u} &= f(\mathcal{S}) = -h(I - SS^T) + \alpha\mathbf{n} \\ \mathbf{v} &= g(\mathbf{q}) = \gamma\left(h(\mathbf{q}\mathbf{q}^T) + \beta\mathbf{n}\right), \end{aligned} \quad (2)$$

with

$$\begin{aligned} \alpha &= \frac{d-k}{d\sqrt{2}} \\ \beta &= -\frac{\|\mathbf{q}\|^2}{d\sqrt{2}} \\ \gamma &= \frac{1}{\|\mathbf{q}\|^2}\sqrt{\frac{k(d-k)}{d-1}}. \end{aligned} \quad (3)$$

We show this construction satisfies the following claim.

Claim 2.1 *$\|\mathbf{v} - \mathbf{u}\|^2 = \mu\,\mathrm{dist}^2(\mathbf{q}, \mathcal{S}) + \nu$, where the constants $\mu = \gamma > 0$ and $\nu \geq 0$ satisfies*

$$\nu = \left(1 - \frac{k}{d}\right)\left(k - \sqrt{\frac{k(d-k)}{d-1}}\right)$$

In particular, $\mu = 1/\|q\|^2$ and $\nu = 0$ when $k = 1$ for all d, and $\mu \approx \sqrt{k}/\|q\|^2$ and $\nu \approx k - \sqrt{k}$ when $k \ll d$.

Affine Subspaces: We represent a k dimensional affine subspace $\mathcal{A}$ by a $(d+1) \times (d-k)$ matrix $\hat{Z}$ whose first d rows contain orthonormal columns, representing the space orthogonal to $\mathcal{A}$, and last row contains a vector of offsets. We further represent the query by homogeneous coordinates, $\hat{\mathbf{q}} = (\mathbf{q}^T, 1)^T$. Our transformations map $\mathcal{A}$ and $\hat{\mathbf{q}}$ to $\mathcal{R}^{\hat{d}'}$, where now $\hat{d}' = (d+1)(d+2)/2 + 1$. Finally, using the $(d+1) \times (d+1)$ matrix $I_A = \mathrm{diag}\{1, ...1, 0\}$, we denote by $\hat{\mathbf{n}} = (\sqrt{2}h(I_A), 0) \in \mathcal{R}^{d'}$. We define our mapping as follows:

$$\begin{aligned} \hat{\mathbf{u}} &= \hat{f}(\mathcal{A}) = -(h(\hat{Z}\hat{Z}^T), \hat{c}(\mathcal{A})) + \hat{\alpha}\hat{\mathbf{n}} \\ \hat{\mathbf{v}} &= \hat{g}(\mathbf{q}) = \hat{\gamma}\left((h(\hat{\mathbf{q}}\hat{\mathbf{q}}^T), 0) + \hat{\beta}\hat{\mathbf{n}}\right). \end{aligned} \quad (4)$$

with

$$\begin{aligned} \hat{c}(\mathcal{A}) &= \sqrt{\frac{M^4 - \|\hat{Z}\hat{Z}^T\|_{fro}^2}{2}} \\ \hat{\alpha} &= \frac{d-k}{d\sqrt{2}} \\ \hat{\beta} &= -\frac{\|\mathbf{q}\|^2}{d\sqrt{2}} \\ \hat{\gamma} &= \frac{1}{\|\mathbf{q}\|^2}\sqrt{\frac{dM^4 - (d-k)^2}{d-1}}, \end{aligned} \quad (5)$$

with a sufficiently large constant M (see Section 2.3). We show this construction satisfies the following claim:

Claim 2.2 *$\|\hat{\mathbf{v}} - \hat{\mathbf{u}}\|^2 = \hat{\mu}\,\mathrm{dist}^2(\mathbf{q}, \mathcal{A}) + \hat{\nu}$, with constants $\hat{\mu} > 0$ and $\hat{\nu} \geq 0$.*

The remainder of this section provides a detailed derivation of these claims. We begin by focusing on the case of linear subspaces (Section 2.1) and investigate the properties of our derivation (Section 2.2). Later on (Section 2.3) we extend this derivation to the case of affine subspaces.

2.1. Linear subspaces

Our derivation is based on the relation between inner products and norms in Euclidean spaces. We first show that the squared distance between a point $\mathbf{q}$ and a subspace $\mathcal{S}$, $\mathrm{dist}^2(\mathbf{q}, \mathcal{S})$, can be written as an inner product, and then that the vectors obtained in this derivation have constant norms. This leads to a basic derivation, which we later modify in Section 2.2 to achieve the final proof of Claim 2.1. Let S be a $d \times k$ matrix whose columns form an orthonormal basis for $\mathcal{S}$, and let Z be a $d \times (d-k)$ matrix whose columns form an orthonormal basis to the null space of S. The distance between $\mathbf{q}$ and $\mathcal{S}$ is given by $\mathrm{dist}(\mathbf{q}, \mathcal{S}) = \|ZZ^T\mathbf{q}\|$, where $ZZ^T\mathbf{q}$ is the projection of $\mathbf{q}$ onto the columns of Z. Since ZZ^T is symmetric and $Z^TZ = I$, implying that $(ZZ^T)^T(ZZ^T) = ZZ^T$, we obtain:

$$\mathrm{dist}^2(\mathbf{q}, \mathcal{S}) = \mathbf{q}^T ZZ^T \mathbf{q}. \quad (6)$$

This can be written as

$$\mathbf{q}^T ZZ^T \mathbf{q} = \sum_{i=1}^{d}\sum_{j=1}^{d}[ZZ^T \otimes \mathbf{q}\mathbf{q}^T]_{ij}, \quad (7)$$

where we use the symbol '$\otimes$' to denote the Hadamard (element-wise) product of two matrices. In other words, if

we rearrange the elements of the matrices ZZ^T and $\mathbf{q}\mathbf{q}^T$ as two vectors in $\mathcal{R}^{d^2}$ then the squared distance can be written as an inner product between those vectors.

Exploiting the symmetry of both ZZ^T and $\mathbf{q}\mathbf{q}^T$ we can embed the two vectors in $\mathcal{R}^{d'}$ with $d' = d(d+1)/2$. This can be done using the operator $h(.)$ defined earlier in (1):

$$\begin{aligned} \bar{\mathbf{u}} &= -h(ZZ^T) \in \mathcal{R}^{d'} \\ \bar{\mathbf{v}} &= h(\mathbf{q}\mathbf{q}^T) \in \mathcal{R}^{d'}. \end{aligned} \quad (8)$$

Note, that $ZZ^T = I - SS^T$, and so it is unnecessary to explicitly construct a basis for the null space. It can now be readily verified that

$$\bar{\mathbf{u}}^T\bar{\mathbf{v}} = -\frac{1}{2}\sum_{ij}[ZZ^T \otimes \mathbf{q}\mathbf{q}^T] = -\frac{1}{2}\mathrm{dist}^2(\mathbf{q}, S). \quad (9)$$

Therefore,

$$\|\bar{\mathbf{u}} - \bar{\mathbf{v}}\|^2 = \|\bar{\mathbf{u}}\|^2 + \|\bar{\mathbf{v}}\|^2 + \mathrm{dist}^2(\mathbf{q}, S), \quad (10)$$

and it is left to show that both $\|\bar{\mathbf{u}}\|$ and $\|\bar{\mathbf{v}}\|$ are constants. Next we show that $\|\bar{\mathbf{u}}\|$ is constant for *all* subspaces of a given intrinsic dimension k, while $\|\bar{\mathbf{v}}\|$ varies with the query. To see this notice that

$$\|\bar{\mathbf{u}}\|^2 = (1/2)\|ZZ^T\|^2_{Fro}, \quad (11)$$

where $\|.\|_{Fro}$ denotes the Frobeneous norm, defined as $\|ZZ^T\|^2_{Fro} = \mathrm{Tr}(ZZ^T(ZZ^T)^T)$, and $\mathrm{Tr}(.)$ denotes the trace of a matrix. Using the identity $ZZ^T(ZZ^T)^T = ZZ^T$ and the properties of the trace we obtain $\mathrm{Tr}(ZZ^T) = \mathrm{Tr}(Z^TZ) = \mathrm{Tr}(I) = d - k$, and so $\|\bar{\mathbf{u}}\|^2 = (1/2)(d-k)$, implying that our transformation maps each subspace in the database to a point on the surface of a sphere. Similarly, $\|\bar{\mathbf{v}}\|^2 = (1/2)\|\mathbf{q}\mathbf{q}^T\|^2_{Fro} = (1/2)\|\mathbf{q}\|^4$ is a constant that depends on $\mathbf{q}$.

In summary, we found a mapping $\bar{\mathbf{u}} = \bar{f}(\mathcal{S})$ and $\bar{\mathbf{v}} = \bar{g}(\mathbf{q})$ which satisfies

$$\|\bar{\mathbf{u}} - \bar{\mathbf{v}}\|^2 = \mathrm{dist}^2(\mathbf{q}, \mathcal{S}) + \frac{1}{2}(d - k + \|\mathbf{q}\|^4), \quad (12)$$

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

Subspaces of varying intrinsic dimension: When the database $\{\mathcal{S}^1, , ..., \mathcal{S}^n\}$ contains subspaces with different intrinsic dimension $k_1, ..., k_n$, we obtain a different norm for each: $\|\bar{\mathbf{u}}^i\|^2 = (1/2)(d - k_i)$. We can handle this by introducing an additional entry to each $\bar{\mathbf{u}}^i$ as follows:

$$\begin{aligned} \breve{\mathbf{u}}^i &= -(h(ZZ^T), \breve{c}(\mathcal{S}^i)) \in \mathcal{R}^{d'+1} \\ \breve{\mathbf{v}} &= (h(\mathbf{q}\mathbf{q}^T), 0) \in \mathcal{R}^{d'+1}. \end{aligned} \quad (13)$$

with $\breve{c}(\mathcal{S}^i)) = \sqrt{(1/2)(k_i - k_{\min})}$, where $k_{\min}$ is the dimension of the thinnest subspace in the database. Note, that the additional entry does not change $\|\bar{\mathbf{v}}\|$ or the inner product $\bar{\mathbf{u}}^T\bar{\mathbf{v}}$.

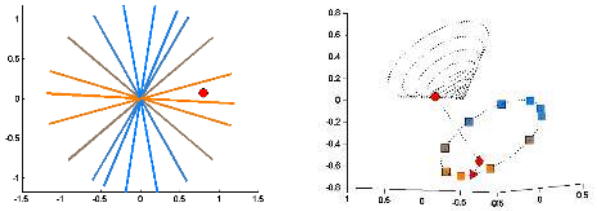


Figure 1. **The geometry of our mapping.** Example of 1D subspaces in $\mathcal{R}^2$, color coded according to distance from a query (left) and their mapping (right). In the basic construction (8) the database lines and potential queries are mapped respectively to a ring and a cone in $\mathcal{R}^3$. The figure shows the query mapped to the cone, then projected to the hyperplane and scaled to lie on the ring.

2.2. The geometry of the mapped database

The transformations introduced earlier in Section 2.1 map all the linear subspaces of dimension k to points on the surface of a sphere of radius $\sqrt{(d-k)/2}$ in $\mathcal{R}^{d'}$. Further inspection of this mapping reveals that all these points also lie on an affine hyperplane. This is a consequence of $\mathrm{Tr}(ZZ^T) = d - k$, which implies that the sum of the components of $\bar{\mathbf{u}}$ that correspond to the diagonal element of ZZ^T is constant. At the same time query points are mapped to the surface of a spherical cone, whose apex is in the origin and whose main axis is orthogonal to this hyperplane. We can use these facts to further project the query to the same hyperplane, and thus reduce the additive constant introduced to the distances by our mapping. This is illustrated in Figure 1, which shows an example of a database of 1D linear subspaces in 2D and their mapping to points in 3D. As is discussed later in Section 3 this will further improve the performance of the nearest neighbor search.

Using the vector $\mathbf{n}$ defined in the beginning of Section 2 we can express the hyperplane by the following formula

$$-\sqrt{2}\mathbf{n}^T\bar{\mathbf{u}} = d - k. \quad (14)$$

We can shift this hyperplane so that it goes through the origin by setting $\mathbf{u} = \bar{\mathbf{u}} + \alpha\mathbf{n}$ with $\alpha = (d-k)/(d\sqrt{2})$. The hyperplane after this translation is then given by $\mathbf{n}^T\mathbf{u} = 0$.

Given a query $\mathbf{q}$ and its mapped version $\bar{\mathbf{v}}$ we seek to project $\bar{\mathbf{v}}$ onto this translated hyperplane. That is, we seek a scalar β such that $\mathbf{v} = \bar{\mathbf{v}} + \beta\mathbf{n}$ lies on the hyperplane:

$$\mathbf{n}^T(\bar{\mathbf{v}} + \beta\mathbf{n}) = 0. \quad (15)$$

Notice that $\sqrt{2}\mathbf{n}^T\bar{\mathbf{v}} = \|\mathbf{q}\|^2$ and $\mathbf{n}^T\mathbf{n} = d$, and so

$$\|\mathbf{q}\|^2 + \sqrt{2}d\beta = 0, \quad (16)$$

from which we obtain $\beta = -\|\mathbf{q}\|^2/(d\sqrt{2})$. Therefore, we can map the query point $\mathbf{q}$ to $\mathbf{v} = \bar{\mathbf{v}} + \beta\mathbf{n}$ and by this reduce the additive constant in (12) by $(\|q\|^2 + d - k)^2/(2d)$.

By uniformly scaling the query point $\mathbf{q}$ we can bring it even closer to the mapped database. Note that uniform

scaling of $\mathbf{q}$ maintains the monotonicity of the mapping. Specifically, we can scale $\mathbf{q}$ such that its norm after scaling becomes $((d-k)k/(d-1))^{1/4}$. Then, after mapping and projection the query will fall on the intersection of the database sphere and its hyperplane. The obtained squared distances after these transformations will be related *linearly* to the original squared distances by $\mu\,\mathrm{dist}^2(q,\mathcal{S})+\nu$ with the constants μ and ν as given in Claim 2.1. Interestingly, in the case of subspaces of rank 1 $\nu=0$, and so if $\mathbf{q}$ lies on $\mathcal{S}$ then $\mathbf{u}$ and $\mathbf{v}$ coincide regardless of d. For subspaces of higher rank ($1<k<d$) $\nu>0$ and so $\mathbf{u}$ and $\mathbf{v}$ cannot coincide because the subspaces only sparsely occupy the sphere.

Note, that the case of subspaces of varying dimension is somewhat more complicated since in this case the mapped subspaces lie on parallel hyperplanes according to their intrinsic dimension. In this case the query can only be projected to the nearest hyperplane.

2.3. Affine subspaces

With few modifications similar transformations can be derived for databases containing a collection of affine spaces. An affine subspace $\mathcal{A}$ is represented by a linear subspace $\mathcal{S}$, provided as a $d\times k$ matrix S with orthonormal columns (or by its null space, provided as a $d\times(d-k)$ matrix Z) 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 $\mathbf{t}^T$. Given a query $\mathbf{q}\in\mathcal{R}^d$ we use homogenous coordinates, denoting $\hat{\mathbf{q}}=(\mathbf{q}^T,1)^T$.

We define a mapping similar to that of linear spaces, this time using $\hat{Z}$ and $\hat{\mathbf{q}}$ instead. The columns of $\hat{Z}$ are not orthonormal due to the additional last row. To account for this we will need to slightly modify our mapping, as follows.

$$\begin{aligned}\tilde{\mathbf{u}} &= \hat{f}(\mathcal{A})=-(h(\hat{Z}\hat{Z}^T),\hat{c}(\mathcal{A}))\in\mathcal{R}^{\hat{d}'}\\ \tilde{\mathbf{v}} &= \hat{g}(\mathbf{q})=(h(\hat{\mathbf{q}}\hat{\mathbf{q}}^T),0)\in\mathcal{R}^{\hat{d}'}.\end{aligned}\tag{17}$$

$\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ 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 $\tilde{\mathbf{u}}$ equal across the database. To achieve this we set $\hat{c}(\mathcal{A})=\sqrt{(M^4-\|\hat{Z}\hat{Z}^T\|_{fro}^2)/2}$, where $\|\hat{Z}\hat{Z}^T\|_{fro}^2=\|ZZ^T\|_{fro}^2+2\|Z\mathbf{t}\|^2+\|\mathbf{t}\|^2=d-k+3\|\mathbf{t}\|^2$ 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 $\tilde{\mathbf{v}}$ to zero, so that the last entry of $\tilde{\mathbf{u}}$ does not affect the inner product of $\tilde{\mathbf{u}}^T\tilde{\mathbf{v}}$. Consequently, $\|\tilde{\mathbf{u}}\|^2=(1/2)M^4$, $\|\tilde{\mathbf{v}}\|^2=(1/2)\|\hat{\mathbf{q}}\|^4$, and $\tilde{\mathbf{u}}^T\tilde{\mathbf{v}}=-\frac{1}{2}\sum_{ij}[\hat{Z}\hat{Z}^T\otimes\hat{\mathbf{q}}\hat{\mathbf{q}}^T]=-\frac{1}{2}\mathrm{dist}^2(\mathbf{q},\mathcal{A})$, and we obtain

$$\|\tilde{\mathbf{u}}-\tilde{\mathbf{v}}\|^2=\mathrm{dist}^2(\mathbf{q},\mathcal{A})+\frac{1}{2}(M^4+\|\hat{\mathbf{q}}\|^4),\tag{18}$$

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

Similar 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 $-\sqrt{2}\mathbf{n}^T\tilde{\mathbf{u}}=d-k$, and so the query can be projected into this hyperplane in the same way as in Section 2.2, yielding the result stated in Claim 2.2.

3. Nearest Neighbor Search

Once the subspaces in the database are mapped to points we can find the nearest subspace to a query by applying a nearest neighbor search for points. Naturally, we wish to employ an efficient solution to this problem. The problem of nearest neighbor search has been investigated extensively in recent years, and algorithms for solving both the exact and approximate versions of the problem exist. For example, for a database containing n points in $\mathcal{R}^d$, exact nearest neighbor can be found by transforming the problem to a point location problem in an arrangement of hyperplanes [1], which can in turn be solved using Meiser's ray shooting algorithm in time $O(d^5\log n)$ [14]. This algorithm, however, requires preprocessing time and storage of $O(n^{d+1})$, which may be prohibitive for typical vision applications.

Approximate solutions to the nearest neighbor problem can achieve comparable query times using a linear ($O(dn)$) preprocessing time and storage. These algorithms return, given a query and a specified constant $\epsilon>0$, 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. Tree based techniques [2] perform this task using at most $O(d^{d+1}\epsilon^{-d}\log n)$, and despite the exponential term they appear to run much faster than a sequential scan even in fairly high dimensions. Another popular algorithm is the locality sensitive hashing (LSH) [8, 11]. LSH is designed to solve the *near neighbor* problem, in which given r and ϵ we seek a neighbor of distance at most $r(1+\epsilon)$ from the query, provided the nearest neighbor lies within distance r from the query. LSH finds a near neighbor in $O(dn^{1/(1+\epsilon)}\log n)$ operation. An approximate nearest neighbor can then be found using an additional binary search on r, increasing the overall runtime complexity by a $O(\log n/\epsilon)$ factor.

Both tree search and LSH provide attractive ways for solving the nearest subspace problem by applying them after mapping. However, we should take notice of two issues. First, our formulation maps subspaces of dimension d to points of dimension $d'=O(d^2)$. In many vision applications this may be intolerably large. Second, the mapping increases the distances from a query to items in the database linearly, with a constant offset that may be large, particularly when the nearest affine subspace is sought. Therefore, to guarantee finding an answer that is not too far from the nearest neighbor we may need to use a significantly smaller

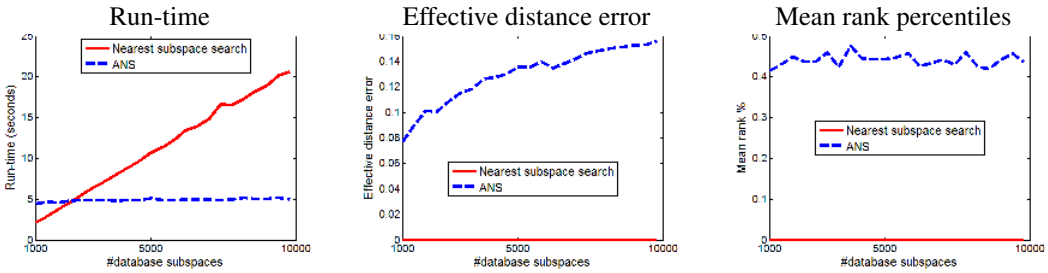


Figure 2. **Varying database size n.** Comparing ANS with nearest subspace search, where ambient dimension is $d = 60$ and intrinsic dimension is $k = 4$.

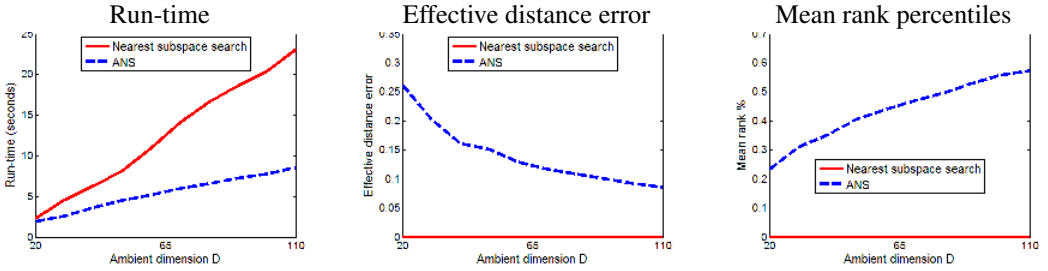


Figure 3. **Varying ambient dimension d.** Comparing ANS with nearest subspace search, where database size is $n = 5000$ and intrinsic dimension is $k = 4$.

ϵ. In particular, suppose we expect the nearest neighbor to lie at some distance t from a query, and denote by $\mu t^2 + \nu$ the corresponding squared distance produced by our mapping. Let $a = \nu/t^2$, then to obtain an approximation ratio of $1 + \epsilon$ in the original space $\mathcal{R}^d$ we would need to select a ratio $1 + \epsilon' \approx (1 + \epsilon)/\sqrt{\mu + a}$ in the mapped space $\mathcal{R}^{d'}$, which can be very small particularly if we expect t to be small. Both issues can lead to a significant degradation in the performance of the nearest neighbor algorithm.

We would like to note further that when $k > 1$ the non-zero constant offset ν is in fact inherent to our method. Using simple arguments it can be shown that there exists no reduction of the nearest subspace problem to nearest neighbor with points such that $\nu = 0$, except for the trivial mapping. Specifically, if $\nu = 0$ any query point that is incident to a subspace, and consequently any two subspaces with non-trivial intersection, must be mapped to the same point. As there exists a chain of intersections between any two subspaces, the only possible mapping in the case that $\nu = 0$ is the trivial mapping.

We approach these problems 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 [12] 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 [15] (see also [16]) 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 either 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.

3.1. Algorithmic details

Given a database subspace we apply the mapping (2) if the subspace is linear, or (4) for affine subspaces. We then preprocess the database as required by our choice of ANN scheme (e.g., build kd-trees, or hash tables). Our preprocessing thus requires the same amount of space as would be used by the selected ANN method, on a database of points in $O(d^2)$.

Given a query our search proceeds by mapping it using (2) or (4), based on a search for linear or affine subspaces. We then call our selected ANN method to report a database point near to our query. Our query running time is thus $O(kd^2) + T_{ANN}(n, d^2)$, where $O(kd^2)$ is the time required for mapping, and $T_{ANN}(n, d)$ is the running time for a choice of an ANN method (e.g., kd-trees, LSH), on a set of n points in $\mathcal{R}^d$.

In the experiments reported below, we have 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. On each random projection we extract c approximate nearest neighbors. Finally, we compute the true distance between the query and all cN_P candidates, and report the closest match across all projections. Our overall query running time is $N_P(O(bd)+O(kb^2)+T_{ANN}(n,b^2)+cO(dk))$, where $O(bd)$ 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.

4. Experiments

We applied our ANS scheme to both synthetic and real data. Run times were measured on a P4 2.8GHz PC with 2GB of RAM (thus, data was loaded to memory in its entirety). Our implementation is in C and uses the ANN kd-tree code of [2], with requested $\epsilon = 100$. We expect similar results when using the LSH scheme. For all our matrix routines we used the OpenCV library. For all our ANS experiments we chose to first project the data to randomly selected subspaces of dimension $b = k + 1$, and then map the projected subspaces to points.

Synthetic data. Figs. 2 and 3 compare run-times and quality of our ANS scheme and sequential subspace search. In Fig. 2 we vary the number of database subspaces, and in Fig. 3 the dimension d. Each test was performed three times with $N_Q = 1000$ queries. For stability we report the median result. 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.

Subspaces were selected uniformly, at random. Following [22], we generate queries such that at least one database subspace is at a distance of no more than $(1+\epsilon)2R\sqrt{(d)}$ from each query, where $R = 0.1$ and $\epsilon = 0.0001$.

Match quality was measured in two ways. First, the effective distance error [2, 13], defined as $Err = (1/N_Q)\sum_q (Dist'/Dist^* - 1)$, where $Dist'$ is the distance from query q to the subspace selected by our algorithm, and $Dist^*$ is the distance between q and its true nearest subspace, computed off line. In addition, we present the mean rank percentile (MRP) for each query, measuring what percentage of the database is closer to the query than the subspace selected by our algorithm.

The results show that our algorithm is faster than sequential database scan, while maintaining fairly low Err rates. In addition, the MRP remains largely robust to the database size n, increasing only moderately in larger dimensions.

Image approximation. We next demonstrate the use of subspaces to represent 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 [6] and reconstruction [10].

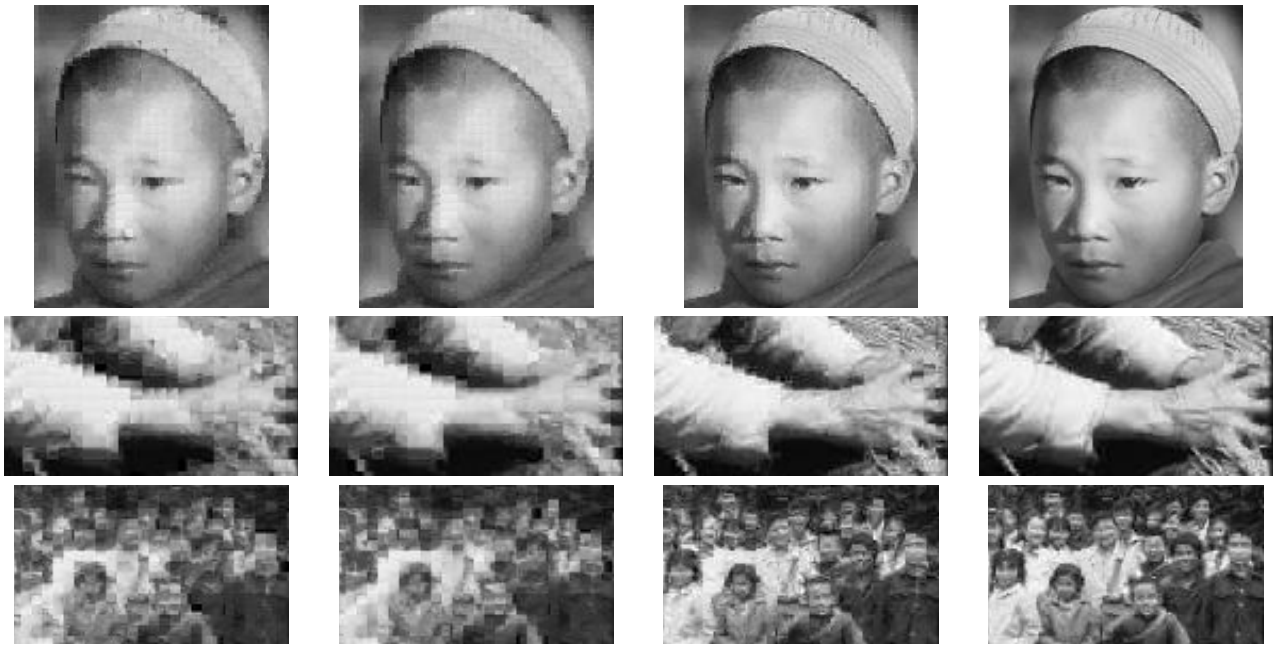
A 1000 random coordinates were selected in a single image (Fig. 5). Then, 16 different, overlapping 5×5 patches around each coordinate were used to produce a $k = 4$ subspace by taking their 4 principal components. These were stored in our *subspace* database. In addition, all 16 patches were stored for our *point* (patch) database.

Given a novel test image we subdivided it into a grid of non-overlapping 5×5 patches. For each patch we searched the point database for a similar patch using (exact) sequential and point ANN based search. The selected database patch was then used as an approximation to the original input patch. Similarly, we used both (exact) sequential and ANS searches to select 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. 4 presents the results obtained by each of the methods. The full images and additional results are included in the supplemental material. 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.

Yale faces. We tested the performance of our ANS scheme on a face recognition task, using faces under varying illumination. In our experiments we used the YaleB face database [9], with images scaled down by a factor of 10. We ran leave-one-out tests, taking one out of the 65 illuminations as a query, and using the rest to produce a database with intrinsic dimension $k = 9$ (following [4, 18]). With 10 subjects and 9 poses, the database is too small to provide the ANS method with a running time advantage over sequential scan (see Fig. 2). However, a comparison of the accuracy of the two methods is reported in Fig. 6. These results imply that although an approximate method makes more mistakes identifying the correct pose and subject, it is comparable to sequential search in detecting either the correct pose or subject.

Yale patches. Motivated by methods using patch data for detection and recognition (e.g. [17]), we tested the capabilities of the ANS method when searching for matching patches sampled from extremely different illumination conditions. For each subject+pose combination (90 altogether) in the YaleB database, we located 50 interest points. We then extracted, from all illuminations, the 9×9 patches centered on each point. Patches from 19 roughly frontal illuminations, were then stored in a *point* database, containing a total of $90 \times 50 \times 19 = 85500$ patches. These same patches



Approx nearest patch | Nearest patch | Approx nearest subspace | Nearest subspace

Figure 4. **Image reconstruction results.** Reconstructed using a single outdoor scene image. See Fig. 5 for run-times and error rates. Full images and more reconstructions included in the supplemental material.

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.

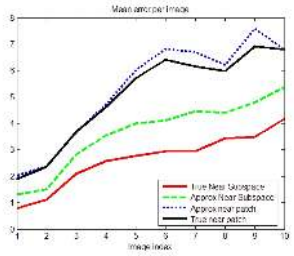


Figure 5. **Image reconstruction details.** From left to right, the single database image used for reconstruction, mean run times for each method, and L1 error reconstruction error. Both database and query images were taken from the Corel data set.

were further used to produce a *subspace* database, containing subspaces of dimension $k = 9$, by taking the nine principal components of sets of corresponding 19 patches. A total of 4500 database subspaces were thus collected. Of the remaining illuminations, we took patches from the most extreme, to be our queries. Fig. 7 presents examples of illuminations used to produce the database and queries.

We evaluate performance as the ability to locate a database item originating from the same semantic part as the query (e.g., eye, mouth etc.) We took a random selection of 1000 query patches, all from the same pose. We then search both point and subspace databases for matches, using exact and approximate nearest neighbor (subspace) methods. Each database item matched with a query then votes for the location of the face center by computing $(c_x, c_y) = (q_x, q_y) + (db_{dx}, db_{dy})$, where (c_x, c_y) is the estimated center of mass, (q_x, q_y) is the position of the query and (db_{dx}, db_{dy}) is the position of the selected database item, relative to its image's center (using cropped images, where the face center is located in the center of the image).

The vote histograms computed by each method are presented in Fig. 7. Under the extreme lighting conditions used, point based methods failed completely since there are no good near neighbors in the data. Both subspace methods managed to locate the correct center of mass, where exact scan did significantly better, but at a higher running time.

5. Conclusion

The advantages of subspaces in pattern recognition and machine vision have been demonstrated time and again. In this paper we have presented a method which facilitates harnessing subspaces, and extending their use to applications involving large scale databases of high dimensions. To this end we have provided an algorithm for sub-linear approx-

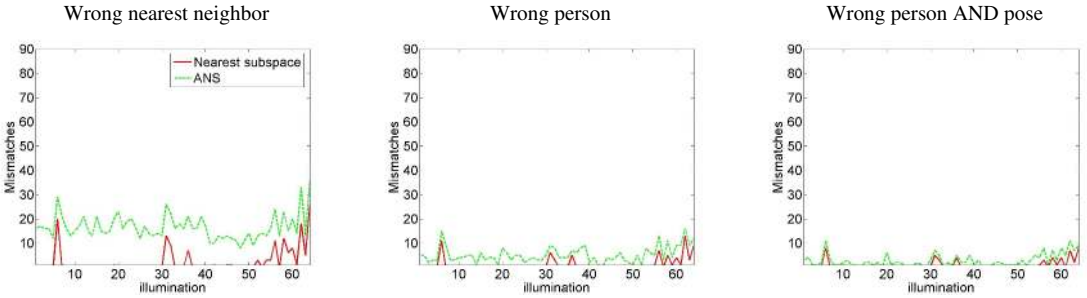


Figure 6. **Faces under varying illumination.** Comparison of face recognition results on the Yale-B face database [9] between exact nearest subspace search and the proposed approximate search. (Left) The approximated search has more errors than exact search. (Middle) The number of times a wrong person was detected by ANS is small and shows that in many cases the nearest neighbor returned by ANS is of the same person in a wrong pose. (Right) In other cases a different person in the correct pose was detected. The frequency at which both the wrong person and the wrong pose were detected is comparable to that of exact search.

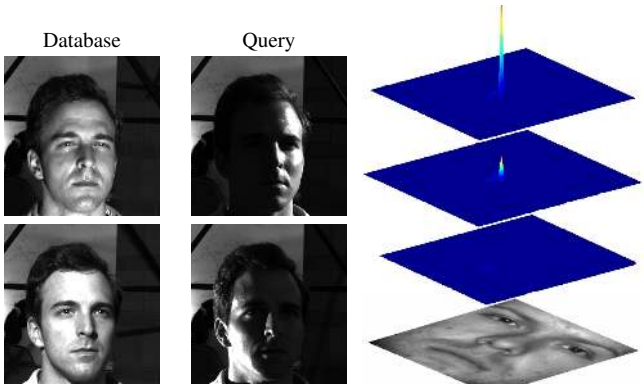


Figure 7. **Yale patches.** (a) Two examples of images used for the databases. (b) Two examples of images used for query patches. Queries were produced from images under extreme illuminations, simulating difficult viewing conditions. (c) Histograms of face-center votes by each selected database item. From top to bottom: Nearest subspace, our ANS method, and nearest patch. These demonstrate the number of times the correct *semantic* item (e.g., eye, mouth) was selected from the database by each method. Run-times were 16.3 seconds for exact subspace search, 11.0 seconds for our ANS method, and 9.2 seconds for the point search.

imate nearest subspace search. Our goal is now to further this study, by investigating different applications which might benefit from these improvement in quality and speed.

References

[1] P.K. Agarwal, J. Erickson, "Geometric range searching and its relatives," *Contemporary Mathematics*, **223**: 1–56, 1999.

[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, D. Jacobs, "Lambertian reflectances and linear subspaces," *IEEE TPAMI*, **25**(2): 218–233, 2003.

[5] V. Blanz, T. Vetter, "Face Recognition based on Fitting a 3D Morphable Model," *IEEE TPAMI*, **25**(9): 1063–1074, 2003.

[6] E. Borenstein, S. Ullman, "Learning to Segment," *ECCV*, **3023**: 315–328, 2004.

[7] M.E. Brand, "Morphable 3D models from video," *CVPR*,**2**: 456–463, 2001.

[8] M. Datar, N. Immorlica P. Indyk, V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*: 253–262, 2004.

[9] 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.

[10] T. Hassner, R. Basri, "Example based 3D reconstruction from single 2D images," In *Beyond Patches workshop, CVPR.*, 2006.

[11] P. Indyk, R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," In *STOC'98*: 604–613, 1998.

[12] W. Johnson, J. Lindenstrauss, "Extensions of Lipschhitz maps into a Hilbert space," *Contemporary Math*: 189–206, **26**, 1984.

[13] T. Liu, A.W. Moore, A. Gray, K. Yang, "An Investigation of Practical Approximate Nearest Neighbor Algorithms," *NIPS*: 825–832, 2004.

[14] S. Meiser, "Point location in arrangements of hyperplanes," *Information and Computation*, **106**: 286–303, 1993.

[15] 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.

[16] A. Naor, P. Indyk, "Nearest neighbor preserving embeddings,", *ACM Transactions on Algorithms*, forthcoming.

[17] L. Fei-Fei, R. Fergus, P. Perona, "One-Shot Learning of Object Categories." *IEEE TPAMI*, **28**(4): 594–611, 2006.

[18] 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.

[19] C. Tomasi, T. Kanade, "Shape and Motion from Image Streams under Orthography: A Factorization Method," *IJCV*, **9**(2): 137–154, 1992.

[20] L. Torresani, D. Yang, G. Alexander, C. Bregler, "Tracking and Modeling Non-Rigid Objects with Rank Constraints," *CVPR*: 493–500, 2001.

[21] S. Ullman, R. Basri, "Recognition by Linear Combinations of Models," *IEEE TPAMI*, **13**(10): 992–1007, 1991.

[22] P.N. Yianilos, "Locally lifting the curse of dimensionality for nearest neighbor search (extended abstract)," *Symposium on Discrete Algorithms*: 361–370, 2000.