Kernel Principal Angles for Classification Machines with Applications to Image Sequence Interpretation*

Lior Wolf Amnon Shashua

School of Computer Science and Engineering
Hebrew University of Jerusalem
Jerusalem, 91904 Israel

Abstract

We consider the problem of learning with instances defined over a space of sets of vectors. We derive a new positive definite kernel $f(A, B)$ defined over pairs of matrices A, B based on the concept of principal angles between two linear subspaces. We show that the principal angles can be recovered using only inner-products between pairs of column vectors of the input matrices thereby allowing the original column vectors of A, B to be mapped onto arbitrarily high-dimensional feature spaces.

We demonstrate the usage of the matrix-based kernel function $f(A, B)$ with experiments on two visual tasks. The first task is the discrimination of "irregular" motion trajectory of an individual or a group of individuals in a video sequence. We use the SVM approach using $f(A, B)$ where an input matrix represents the motion trajectory of a group of individuals over a certain (fixed) time frame. We show that the classification (irregular versus regular) greatly outperforms the conventional representation where all the trajectories form a single vector. The second application is the visual recognition of faces from input video sequences representing head motion and facial expressions where $f(A, B)$ is used to compare two image sequences.

1. Introduction

The paper is about developing a similarity function that operates on pairs of *sets* of vectors — where a vector can represent an image and a set of vectors could represent a video sequence for example — in such a way that the function can be plugged into a variety of existing classification engines. The crucial ingredients are therefore (i) the function can be evaluated in high dimensional spaces using simple functions (kernel functions) evaluated on pairs of vectors in the original (relatively low-dimensional) space, and (ii) the function describes an inner-product space, i.e., is a positive definite kernel.

It would be natural to ask why would one need such a function to begin with? The conventional approach to representing a signal for classification tasks — be it a 2D image, a string of characters or any 1D signal — is to form a one-dimensional attribute vector $\mathbf{x}_i$ in some space R^n defined as the instance space. Whether the instance space is a vector space or not is not really crucial for this discussion, but the point being is that instances are essentially 1-dimensional objects.

However, there are situations which call for representing an instance as a *set* of vectors. For example, in a visual interpretation task the models themselves may be obtained from sets of images (such as a video sequence), and in machine learning when a training set is pre-expanded to contain virtual examples in order to incorporate prior knowledge about invariances of the input data. To be concrete, we will describe three such situations below.

The first situation is a classical face detection problem. Face recognition has been traditionally posed as the problem of identifying a face from a single image. On the other hand, contemporary face tracking systems can provide long sequences of images of a person, thus for better recognition performance it has been argued ([18, 22], for example) that the information from all images should be used in the classification process. One is therefore faced with the problem of matching between two sets of images (where each image is represented by a vector of pixel values).

The second situation is also related to visual interpretation but in a different setting. Consider for example a visual surveillance task of deciding whether a video sequence of people in motion contains an "irregular" trajectory. The application can vary from detection of shop-lifting, breaking-and-entry or the detection of "irregular" movements of an individual in a crowd. Given that the motion trajectory of an individual can be modeled as a vector of positions over time, then the most natural representation of the entire video clip is a *set* of vectors. We would be looking, therefore, for an appropriate set-matching measure which could be plugged-

*This paper should be referenced as "Hebrew University, School of CSE, Technical Report TR-2002-48, Nov. 2002".

in into conventional classification engines. More details are provided in Section 6.

The third situation occurs when conventional classification engines are incorporated with prior knowledge about invariances of the input vectors. By invariances we mean certain transformations which leave class membership invariant. In digit recognition, for example, typical invariances include line thickness and image plane translation and rotation. It has been observed that an effective way to make a classifier invariant is to generate synthetic training examples by transforming them according to the desired invariances (cf. [2, 8, 16, 19]). For instance the "kernel jittering" of [8] performs the synthetic transformations within the matching process between pairs of training examples, thereby effectively matching between two sets of vectors (or between a vector and a set).

We could for convenience represent the collection of vectors in R^n as columns of a matrix, thus our instance space is the space over matrices. In all three examples above, the order of the columns of a training matrix is unimportant, thus the similarity metric over a pair of matrices we wish to derive should ideally match between the two respective column spaces, rather than between the individual columns. Another useful property we desire is to incorporate the similarity metric with a non-linear "feature map" $\phi : R^n \rightarrow \mathcal{F}$ with a corresponding kernel satisfying $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$. A typical example is a feature map of dimension $\binom{n+d-1}{d}$ representing the d'th order monomial expansion of the input vector with the corresponding kernel $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}')^d$. Working with feature maps allows one to represent non-linearities, as for example the linear subspace defined by the column space of the matrix $A = [\phi(\mathbf{a}_1), ..., \phi(\mathbf{a}_k)]$ is a *surface* in the original input space R^n. Therefore, the measure of similarity between two matrices undergoing a feature map translates to a measure between the two underlying surfaces in R^n. Because of the prohibitly high dimension of the feature space, we would not like to ever evaluate the function $\phi()$ thereby the "kernel trick" is possible only if the similarity metric $f(A, B)$ can be implemented using *inner products only* between the columns of $A = [\phi(\mathbf{a}_1), ..., \phi(\mathbf{a}_k)]$ and $B = [\phi(\mathbf{b}_1), ..., \phi(\mathbf{b}_k)]$. Finally, to make general use of the similarity function, we also desire that $f(A, B)$ forms a positive definite kernel on its own accord (for reasons described later).

In this paper we propose a measure over the principal angles between the two column spaces of the input matrices A, B. The principal angles are invariant to the column ordering of the two matrices thereby representing a measure over two unordered sets of vectors. The challenge in this work is two fold: the first challenge is to compute the principal angles in feature space using only inner-products between the columns of the input matrices, i.e., using only computations of the form $k(\mathbf{a}_i, \mathbf{b}_j), k(\mathbf{a}_i, \mathbf{a}_j)$ and $k(\mathbf{b}_i, \mathbf{b}_j)$ for $i, j = 1, ..., k$. The second challenge is to introduce an appropriate function over the principal angles such that $f(A, B)$ forms a positive definite kernel.

1.1 Related Work

The idea of using principal angles as a measure for matching two image sequences was proposed in [22] with dissimilarity between the two subspaces measured by the smallest principal angle — thereby effectively measuring whether the subspaces *intersect* which is somewhat similar to a "nearest neighbor" approach. However, the assumption that a linear subspace is a good representation of the input set of vectors is somewhat restrictive with decreasing effectiveness for low dimension n and large input set size k. In our approach, the dimension of the feature space is very high and due to the use of the kernel trick one effectively matches two non-linear surfaces in R^n instead of linear subspaces.

Another recent approach proposed by [18] to match two image sequences is to compute the covariance matrices of the two input sets and use the Kullback-Leibler divergence metric (algebraically speaking, a function of $AA^\top, BB^\top$ assuming zero mean column spaces) assuming the input set of vectors form a Gaussian distribution. The fact that only input space dimension R^n is used constrains the applicability of the technique to relatively small input sets, and the assumption of a Gaussian distribution limits the kind of variability along the input sequence which can be effectively tolerated.

Other ideas published in the context of matching image sequences are farther away from the concepts we propose in this paper. The common idea in most of the published literature is that recognition performance can be improved by modeling the variability over the input sequence. Most of those ideas are related to capturing "dynamics" and "temporal signatures" [9, 12, 4].

Finally, in the "kernel jittering" [8] approach for obtaining invariances over a class of transformations, two instance vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ are matched by creating additional synthetic examples $\mathbf{x}_{ip}$ and $\mathbf{x}_{jq}$ centered around the original input instances and selecting $k(\mathbf{x}', \mathbf{x}'')$ as the output measure of the two sets based on a nearest neighbor concept. The problem with this approach is that the measure does not necessarily form a positive definite kernel and the nearest neighbor approach is somewhat ad-hoc. In our approach, the two subspaces spanned by $\phi(\mathbf{x}_{ip})$ and $\phi(\mathbf{x}_{jq})$, respectively, would be matched using a positive definite kernel. In Section 5 we will demonstrate the superiority of our similarity measure over sets against a nearest neighbor approach in the context of a jittering experiment.

2 Kernel Principal Angles

Let the columns of $A = [\phi(\mathbf{a}_1), ..., \phi(\mathbf{a}_k)]$ and $B = [\phi(\mathbf{b}_1), ..., \phi(\mathbf{b}_k)]$ represent two linear subspaces U_A, U_B in the feature space where $\phi()$ is some mapping from input space R^n onto a feature space $\mathcal{F}$ with a kernel function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$. The principal angles $0 \leq \theta_1 \leq ... \leq \theta_k \leq (\pi/2)$ between the two subspaces are uniquely defined as:

$$cos(\theta_k) = \max_{\mathbf{u} \in U_A} \max_{\mathbf{v} \in U_B} \mathbf{u}^\top \mathbf{v} \tag{1}$$

subject to:

$$\mathbf{u}^\top \mathbf{u} = \mathbf{v}^\top \mathbf{v} = 1, \quad \mathbf{u}^\top \mathbf{u}_i = 0, \mathbf{v}^\top \mathbf{v}_i = 0, \quad i = 1, ..., k-1$$

The concept of principal angles is due to Jordan in 1875, where [13] is the first to introduce the recursive definition above. The quantities $cos(\theta_i)$ are sometimes referred to as *canonical correlations* of the matrix pair (A, B). There are various ways of formulating this problem, which are all equivalent, but some are more suitable for numerical stability than others. A numerically stable algorithm was proposed by [5] based on the QR factorization and SVD, as follows.

Let $A = Q_A R_A$ and $B = Q_B R_B$ where Q is an orthonormal basis of the respective subspace and R is a upper-diagonal $k \times k$ matrix with the Gram-Schmidt coefficients representing the columns of the original matrix in the new orthonormal basis. The singular values $\sigma_1, ..., \sigma_k$ of the matrix $Q_A^\top Q_B$ are the principal angles $cos(\theta_i) = \sigma_i$.

The challenge of computing the principal angles is that the matrices Q_A, Q_B should never be *explicitly* evaluated because the columns of the Q matrices are in the high dimensional feature space. *Our task therefore is to compute $Q_A^\top Q_B$ without computing the individual matrices Q_A, Q_B.*

Consider the result of the Gram-Schmidt orthogonalization process of the matrix A: Let $\mathbf{v}_j \in \mathcal{F}$ be defined as:

$$\mathbf{v}_j = \phi(\mathbf{a}_j) - \sum_{i=1}^{j-1} \frac{\mathbf{v}_i^\top \phi(\mathbf{a}_j)}{\mathbf{v}_i^\top \mathbf{v}_i} \mathbf{v}_i \tag{2}$$

Let $V_A = [\mathbf{v}_1, ..., \mathbf{v}_k]$ and

$$\mathbf{s}_j = (\frac{\mathbf{v}_1^\top \phi(\mathbf{a}_j)}{\mathbf{v}_1^\top \mathbf{v}_1}, ..., \frac{\mathbf{v}_{j-1}^\top \phi(\mathbf{a}_j)}{\mathbf{v}_{j-1}^\top \mathbf{v}_{j-1}}, 1, 0, 0, ..., 0)^\top \tag{3}$$

Then,

$$A = V_A S_A, \tag{4}$$

where $S_A = [\mathbf{s}_1, ..., \mathbf{s}_k]$ an upper diagonal $k \times k$ matrix. The QR factorization is therefore:

$$A = (V_A D_A^{-1})(D_A S_A), \tag{5}$$

where D_A is a diagonal matrix $D_{ii} = || \mathbf{v}_i ||_2$. Assuming the columns of A are linearly independent (this assumption will be removed later) then S_A^{-1} is well defined, and

$$A = A S_A^{-1} D_A^{-1} D_A S_A, \tag{6}$$

from which we obtain: $Q_A = A R_A^{-1}$ and $R_A = D_A S_A$. The last step taking us from (5) to (6), although may appear somewhat artificial, is actually the key to making the kernel-based computation of principal angles work. All we will need is to compute D_A and S_A^{-1} (both of which are $k \times k$), and likewise D_B, S_B^{-1}. Then, $Q_A^\top Q_B = R_A^{-T} A^\top B R_B^{-1}$, where $A^\top B$ involves only inner products between the columns of A and B and thus can be computed by using the kernel: $(A^\top B)_{ij} = k(\mathbf{a}_i, \mathbf{b}_j)$.

What remains to show is that D_A, S_A^{-1} can be computed with only inner-products of the columns of A. We will describe now an interleaving algorithm for computing the columns $\mathbf{s}_i$ of the matrix S_A and the columns $\mathbf{t}_i$ of S_A^{-1} one at a time.

From (4) we have $V_A = A S_A^{-1}$, thus $\mathbf{v}_j = A \mathbf{t}_j$ and due to the nature of the Gram-Schmidt process (S_A is upper diagonal) we have:

$$\mathbf{v}_j = \sum_{q=1}^{j} t_{qj} \phi(\mathbf{a}_j),$$

where t_{qj} is the q'th element of the vector $\mathbf{t}_j$. The inner products $\mathbf{v}_j^\top \phi(\mathbf{a}_i)$ and $\mathbf{v}_j^\top \mathbf{v}_j$ can be computed via a kernel:

$$\mathbf{v}_j^\top \phi(\mathbf{a}_i) = \sum_{q=1}^{j} t_{qj} k(\mathbf{a}_i, \mathbf{a}_q) \tag{7}$$

$$\mathbf{v}_j^\top \mathbf{v}_j = \sum_{p=1}^{j} \sum_{q=1}^{j} t_{pj} t_{qj} k(\mathbf{a}_p, \mathbf{a}_q) \tag{8}$$

The inner-products above are the building blocks of D_A — whose diagonal consists of the norm of $\mathbf{v}_j$ which is computed via (8). From (3), the columns $\mathbf{s}_j$ of S_A are defined as:

$$\mathbf{s}_j = (\frac{t_{11} k(\mathbf{a}_1, \mathbf{a}_j)}{t_{11}^2 k(\mathbf{a}_1, \mathbf{a}_1)}, ... \frac{\sum_{q=1}^{j-1} t_{qj} k(\mathbf{a}_j, \mathbf{a}_q)}{\sum_{p,q=1}^{j-1} t_{pj} t_{qj} k(\mathbf{a}_p, \mathbf{a}_q)}, 1, 0, ..., 0). \tag{9}$$

We see that the columns $\mathbf{s}_j$ depends on $\mathbf{t}_l$ from $l = 2, ..., j$, and conversely $\mathbf{t}_j$ depends on $\mathbf{s}_j$ as well. However, the way to break the cycle of dependency is by noticing that $\mathbf{t}_j$ can be represented as a function of $\mathbf{t}_1, ..., \mathbf{t}_{j-1}$ and of $\mathbf{s}_j$ as follows. From (2) we have:

$$\mathbf{v}_j = [-\mathbf{v}_1, ..., -\mathbf{v}_{j-1}, \phi(\mathbf{a}_j), 0, ..., 0] \mathbf{s}_j, \tag{10}$$

and since $\mathbf{v}_j = A \mathbf{t}_j$ we have by substitution in (10):

$$\mathbf{v}_j = A[-\mathbf{t}_1, ..., -\mathbf{t}_{j-1}, \mathbf{e}_j, 0, ..., 0] \mathbf{s}_j,$$

where $\mathbf{e}_j$ is defined such that $I = [\mathbf{e}_1, ..., \mathbf{e}_k]$ is the $k \times k$ identity matrix. As a result,

$$\mathbf{t}_j = [-\mathbf{t}_1, ..., -\mathbf{t}_{j-1}, \mathbf{e}_j, 0, ..., 0]\mathbf{s}_j. \quad (11)$$

We have described all the elements of the algorithm for computing the principal angles between A, B using only inner-products between the column vectors. We summarize below the algorithm:

- Given two sets of vectors $\mathbf{a}_i, \mathbf{b}_i$, $i = 1, ..., k$ in R^n, we would like to find the principal angles between the two matrices $A = [\phi(\mathbf{a}_1), ..., \phi(\mathbf{a}_k)]$ and $B = [\phi(\mathbf{b}_1), ..., \phi(\mathbf{b}_k)]$ where $\phi()$ is some high-dimensional mapping with a kernel function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$.
- Compute the $k \times k$ matrix R_A^{-1} as follows:
- Let $\mathbf{s}_1 = \mathbf{t}_1 = \mathbf{e}_1$
- Repeat for $j = 2, ..., k$:
  - Compute $\mathbf{s}_j$ using Equation (9).
  - Compute $\mathbf{t}_j$ using Equation (11).
- Compute the diagonal matrix D_A using Equation (8).
- $R_A^{-1} = [\mathbf{t}_1, ..., \mathbf{t}_k] D_A^{-1}$.
- Follow the previous steps and compute R_B^{-1}.
- Let $M_{ij} = k(\mathbf{a}_i, \mathbf{b}_j)$ be the entries of the $k \times k$ matrix $M = A^\top B$.
- The cosine of the principal angles are the singular values of the matrix $R_A^\top M R_B^{-1}$.

It is worthwhile noting that the two sets of vectors need not be of the same size, i.e., the column spaces of A and B need not be of the equal dimensions. The requirement of equal dimensions is necessary only in the context of obtaining a positive definite kernel from the principal angles — which is the topic of the next section. Finally, note that if the column space of A is not full rank then we can omit those $\mathbf{t}_j$ for which $(D_A)_{jj} = 0$ and obtain R_A^{-1} whose number of columns are equal to the rank of A. Likewise for B.

3 Alternative Formulations

The algorithm above for computing the principal angles is based on "kernalizing" the "QR-SVD" formulation which on one hand is known to be the most numerically stable (in the non-kernel case) and on the other hand is computationally efficient where the most intensive part consisting of a single application of SVD on a $k \times k$ matrix.

For the sake of completeness, in the following two sections we will describe two other approaches for kernalizing the computation of principal angles. The first approach is based on the Lagrange formulation [13] where the principal angles are the generalized eigenvalues of an expanded $2k \times 2k$ matrix. The second approach is based on an eigen-decomposition formulation for generating Q_A and Q_B instead of the QR step. Both approaches are less efficient than the QR-SVD approach where the Lagrange formulation suffers from numerical instability as well.

3.1 The Lagrange Formulation

The original formulation by [13] was based on Lagrange multipliers generating the principal angles as the set of generalized eigenvalues of a block diagonal matrix. The approach suffers from numerical stability issues, however, the extension to computation in feature space is *immediate* as shown next. Problem (1) can written as

$$\max_{\mathbf{x},\mathbf{y}} \{\mathbf{y}^\top B^\top A \mathbf{x}\} \quad s.t. \ \|A\mathbf{x}\| = 1, \ \|B\mathbf{y}\| = 1.$$

The Lagrangian of the problem is:

$$L(\mathbf{x}, \mathbf{y}, \lambda, \mu) = \mathbf{y}^\top B^\top A \mathbf{x} - \lambda(\|A\mathbf{x}\|^2 - 1) - \mu(\|B\mathbf{y}\|^2 - 1).$$

After differentiating with respect to $\mathbf{x}, \mathbf{y}, \lambda, \mu$ we obtain that $\lambda = \mu$ and the following condition should hold:

$$\begin{pmatrix} 0 & B^\top A \\ A^\top B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \end{pmatrix} = \lambda \begin{pmatrix} B^\top B & 0 \\ 0 & A^\top A \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \end{pmatrix},$$

subject to $\mathbf{x}^\top A^\top A \mathbf{x} = \mathbf{y}^\top B^\top B \mathbf{y} = 1$. The generalized eigenvalues $\lambda_1, ..., \lambda_{2k}$ are related to the principal angles by $\lambda_1 = cos(\theta_1), ..., \lambda_k = cos(\theta_k)$, and $\lambda_{k+1} = -cos(\theta_k), ..., \lambda_{2k} = -cos(\theta_1)$. The extension to feature space is immediate since the matrices $A^\top A, B^\top A, B^\top B$ involve only inner products of the columns of A and B. The draw-back of this approach is two-fold: on one hand the generalized eigenvalue problem involves a $2k \times 2k$ system compared to a $k \times k$ system with the SVD approach, and second the solution of a generalized eigenvalue system is prone to numerical instabilities — which were confirmed in our experiments.

3.2 The Eigen-decomposition Approach

The QR-SVD formulation was based on generating an orthonormal basis Q_A, Q_B for the column spaces of A, B respectively followed by the SVD of $Q_A^\top Q_B$ to obtain the singular values. An orthonormal basis for A, for example, can be generated from the eigen-decomposition of $AA^\top$ instead of via a QR decomposition. Kernalizing an eigen-decomposition is immediate, but at a price of efficiency: the

overall process for finding the principal angles will consist of 3 applications of SVD of $k \times k$ matrices instead of a single SVD.

Consider the matrix $A = [\phi(\mathbf{a}_1), ..., \phi(\mathbf{a}_k)]$ and let Q_A be the orthonormal basis of the column space of A as generated by the SVD process of A: $AA^\top = Q_A D_A Q_A^\top$ where D_A is a diagonal matrix containing the square eigenvalues of A and the columns of Q_A are the corresponding eigenvectors. Since $AA^\top$ is not-computable (as the columns of A are in the feature space), consider instead the eigenvectors of the $k \times k$ matrix $A^\top A$ whose entries are $k(\mathbf{a}_i, \mathbf{a}_j)$: $A^\top A = U_A D_A U_A^\top$. The connection between U_A and Q_A is easily established as follows:

$$A^\top A U_A = U_A D_A,$$

followed by pre-multiplication by A:

$$AA^\top (AU_A) = (AU_A) D_A,$$

from which we obtain that AU_A is an orthogonal set (unnormalized eigenvectors). Therefore:

$$Q_A = AU_A D_A^{-\frac{1}{2}}$$

is the orthonormal set of eigenvectors of $AA^\top$ which span the column space of A. The derivation above is a well known "trick" used to compute the Principal Components of a matrix whose number of columns is much smaller than the number of rows (see for example, [20]).

The matrix Q_A is not-computable, but the product:

$$Q_A^\top Q_B = D_A^{-\frac{1}{2}} U_A^\top (A^\top B) U_B D_B^{-\frac{1}{2}}$$

is computable. To conclude, the kernel principal angles based on eigen-decompositions is summarized below:

- Given two sets of vectors $\mathbf{a}_i, \mathbf{b}_i$, $i = 1, ..., k$ in R^n, we would like to find the principal angles between the two matrices $A = [\phi(\mathbf{a}_1), ..., \phi(\mathbf{a}_k)]$ and $B = [\phi(\mathbf{b}_1), ..., \phi(\mathbf{b}_k)]$ where $\phi()$ is some high-dimensional mapping with a kernel function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$.

- Let U_A, D_A be the eigen-decomposition using the SVD formulation $A^\top A U_A = U_A D_A$ and likewise let U_B, D_B be the eigen-decomposition of $B^\top B U_B = U_B D_B$. Note that the entries of $A^\top A$ and $B^\top B$ involve the evaluations of $k(\mathbf{a}_i, \mathbf{a}_j)$ and $k(\mathbf{b}_i, \mathbf{b}_j)$ only.

- Let $M_{ij} = k(\mathbf{a}_i, \mathbf{b}_j)$ be the entries of the $k \times k$ matrix $M = A^\top B$.

- The cosine of the principal angles are the singular values of the matrix $D_A^{-\frac{1}{2}} U_A^\top M U_B D_B^{-\frac{1}{2}}$.

This algorithm requires between twice to three times the computational resources of the QR based algorithm since it consists of three applications of SVD. Empirical studies we conducted show that the two algorithms have similar numerical stability properties with slight benefit to the QR approach.

4 Making a Positive Definite Kernel

We have shown so far that given two sets of vectors $\mathbf{a}_i, \mathbf{b}_i$, $i = 1, ..., k$ in R^n one can compute $cos(\theta_i)$, the cosine of the principal angles, between the two subspaces $span\{\phi(\mathbf{a}_1), ..., \phi(\mathbf{a}_k)\}$ and $span\{\phi(\mathbf{b}_1), ..., \phi(\mathbf{b}_k)\}$ where $\phi()$ is a high dimensional mapping with kernel $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$ using only computations of the form $k(\mathbf{a}_i, \mathbf{b}_j), k(\mathbf{a}_i, \mathbf{a}_j)$ and $k(\mathbf{b}_i, \mathbf{b}_j)$ for $i, j = 1, ..., k$. In fact, the two sets of vectors may be of different sizes, but for the material discussed in this section we must assume that the column spaces of A, B are of equal dimension.

In this section we address the issue of constructing a positive definite kernel $f(A, B)$ and consider a number of candidate functions. Specifically, we propose and prove that

$$\Pi_{i=1}^k cos(\theta_i)^2$$

is a positive definite kernel. The reason we would like a similarity measure that can be described by an inner-product space is for making it generally applicable to a wide family of classification and clustering tools. Existing kernel algorithms like the Support Vector Machine (SVM) and "kernel-PCA" (to mention a few) rely on the use of a positive definite kernel to replace the inner-products among the input vectors. Our measure $f(A, B)$ can be "plugged-in" as a *kernel function* provided that for any set of matrices A_i, $i = 1, ..., m$ and for any (positive) integer m, the $m \times m$ matrix K:

$$K_{ij} = f(A_i, A_j)$$

is (semi) positive definite, i.e., $\mathbf{x}^\top K \mathbf{x} \geq 0$ for all vectors $\mathbf{x} \in R^m$. This property enhances the usefulness of $f(A, B)$ for a wider variety of applications, and in some applications (like optimal margin algorithms) it is a necessary condition.

To avoid confusion, the computation of $cos(\theta_i)$ involves the use of some kernel function as was described in the previous section — but this does not necessarily imply that any function $d(\theta_i)$ of $cos(\phi_i)$ is a positive definite kernel, i.e., that there exist some canonical mapping $\psi(A)$ from the space of matrices to a vector space such that $d(\theta_1, .., \theta_k) = \psi(A)^\top \psi(B)$. The result we will need for the remainder of this section is the Binet-Cauchy theorem on the product of compound matrices ([1],pp.93) attributed by Binet and Cauchy in 1812 — described next.

Definition 1 (Compound Matrices) *Let A be an $n \times k$ matrix. The matrix whose elements are the minors of A of order q constructed in a lexicographic order is called the "q'th compound of A" and is denoted by $C_q(A)$.*

In other words, the q'th order minors are the determinants of the sub-matrices constructed by choosing q rows and q columns from A, thus $C_q(A)$ has $\binom{n}{q}$ rows and $\binom{k}{q}$ columns. The priority of choosing the rows and columns for the minors is based on a lexicographic order: minors from rows 1,2,4 for example will appear an in an earlier row in $C_q(A)$ than those from 1,2,5 or 1,3,4 or 2,3,4; and likewise for columns. Of particular interest for us is the *Grassman vector* defined below:

Definition 2 (Grassman Vector) *Let A be an $n \times k$ matrix where $n \geq k$. The k'th compound matrix $C_k(A)$ is a vector of dimension $\binom{n}{k}$ called the Grassman vector of A denoted by $\psi(A)$.*

For example, for $n = 4, k = 2$ the two columns of A may represent two points in the 3D projective space and $\psi(A)$ represents the 6 Grassman (Plucker) coordinates of the line spanned by the two points. The Grassman coordinates are invariant (up to scale) to the choice of the two points on the line. In general, the Grassman coordinates represent the subspace spanned by the columns of A invariantly to the choice of points (basis) of the space. The Binet-Cauchy theorem is described next:

Definition 3 (Binet-Cauchy Theorem) *Let A, B be rectangular matrices of size $n \times k$ and $n \times p$, respectively. Then,*

$$C_q(A^\top B) = C_q(A)^\top C_q(B).$$

In other words, the $\binom{k}{q} \times \binom{p}{q}$ matrices $C_q(A^\top B)$ and $C_q(A)^\top C_q(B)$ are element for element identical. Of particular interest to us is the case where $p = k = q$, thus $C_k(A^\top B)$ is a scalar equal to $det(A^\top B)$ (because $A^\top B$ is a $k \times k$ matrix and $\binom{k}{k} = 1$) from which we obtain the following corollary:

Corollary 1 *Let A, B be matrices of size $n \times k$. Then,*

$$det(A^\top B) = \psi(A)^\top \psi(B).$$

As a result, the measure $det(A^\top B)$ is positive definite. Since the entries of $A^\top B$ are the inner-products of the columns of A, B thus the computation can be done in the so called feature space with kernel $k(\mathbf{a}_i, \mathbf{b}_j) = \phi(\mathbf{a}_i)^\top \phi(\mathbf{b}_j)$ where $\phi()$ is the mapping from the original R^n to some high dimensional feature space. However, $det(A^\top B)$ depends on the choice of the columns of A, B rather than on the respective column spaces (as principal angles do), thus is not likely to be a good candidate for a positive definite kernel $f(A, B)$ over pairs of matrices.

The next immediate choice for $f(A, B)$, is $det(Q_A^\top Q_B)$ since from Corollary 1 we have $det(Q_A^\top Q_B) = \psi(Q_A)^\top \psi(Q_B)$. The choice $f(A, B) = det(Q_A^\top Q_B)$ is better than $det(A^\top B)$ because it is invariant to the choice of basis for the respective column spaces of A, B. Since Q_A, Q_B are orthonormal matrices, a change of basis would result in a product with a rotation matrix: $\tilde{Q}_A = Q_A R_1$ and $\tilde{Q}_B = Q_B R_2$ where R_1, R_2 are some rotation matrices. Then

$$det(\tilde{Q}_A^\top \tilde{Q}_B) = det(R_1^\top) det(R_2) det(Q_A^\top Q_B) = det(Q_A^\top Q_B).$$

The problem, however, is that $det(Q_A^\top Q_B)$ can receive both positive and negative values making it a non-ideal candidate for a measure of similarity. For example, by changing the sign of one of the columns of A, results in $det(Q_A^\top Q_B)$ changing sign, yet the respective column spaces have not changed. On the other hand, the absolute value $|det(Q_A^\top Q_B)|$ may not be positive definite (in fact it isn't as one can easily show by creating a counter example). Nevertheless, the product of two positive definite kernels is also a positive definite kernel (see [17] for example), then

$$f(A, B) = det(Q_A^\top Q_B)^2 = \Pi_{i=1}^k cos(\theta_i)^2$$

is our *chosen positive definite kernel function*.

Finally, for purposes of clarity only it may be worthwhile to show the connection between the inner product $\psi(Q_A)^\top \psi(Q_B)$ and the inner product $\psi(A)^\top \psi(B)$:

Theorem 1 *Let A, B be matrices of size $n \times k$ where $n \geq k$. Then,*

$$cos(\psi(A), \psi(B)) = \psi(Q_A)^\top \psi(Q_B).$$

Proof: This is a result related to a theorem by [6] and [15] which follows directly from the Binet-Cauchy theorem, as follows: Let $A = Q_A R_A$ and $B = Q_B R_B$ represent the QR factorization of both matrices. Note that $det(R_A)$ and $det(R_B)$ are positive (using the algorithm in the previous section). From the Binet-Cauchy theorem we have: $\|\psi(Q_A)\| = \|\psi(Q_B)\| = 1$ (because $det(I) = \psi(Q_A^\top Q_A) = \psi(Q_A)^\top \psi(Q_A)$). Likewise, $\psi(A)^\top \psi(A) = det(R_A^\top R_A)$, thus $\|\psi(A)\| = det(R_A)$. Also note that $\psi(Q_A R_A) = det(R_A)\psi(Q_A)$. Then,

$$\begin{aligned} cos(\psi(A), \psi(B)) &= \frac{\psi(A)^\top \psi(B)}{\|\psi(A)\| \cdot \|\psi(B)\|} \\ &= \frac{det(R_A) det(R_b) \psi(Q_A)^\top \psi(Q_B)}{det(R_A) det(R_B)} \\ &= \psi(Q_A)^\top \psi(Q_B) \end{aligned}$$

□

Note that that if the QR factorization does not produce positive determinants for the R components, the theorem

above is defined up to absolute value only. To conclude, among the possible positive definite kernels we have proposed (which can be computed in feature space) the one which makes the most sense is:

$$f(A,B) = det(Q_A^\top Q_B)^2 = \Pi_{i=1}^k cos(\theta_i)^2 \qquad (12)$$

where $cos(\theta_i)$ are computed in feature space according to the Algorithm described in the previous section.

5 A Case Against Nearest Neighbor Approach

In the context of "kernel jittering", where synthetic copies of the input vectors are created in order to simulate invariances of interest, the matching measurement over sets (or over a vector against a set of vectors) is based on the nearest neighbor principle ([8]). In this section we compare the performance, on a toy problem, of such an approach to our positive definite kernel based on principal angles.

Consider the case of generating virtual examples in order to train the classifier to become invariant to line width (in the context of digit recognition). As an example, we are given three example vectors, each representing an image, and we generate two additional vectors per example by artificially thinning and thickening the lines using morphological operators (see Fig. 1). As a result we obtain three sets of vectors with three vectors each. A good matching-over-sets is one which will be invariant to line width. Note that unlike the case of invariance to translation, since these morphological operators are not symmetrical (information is lost) the matching has to be done between two sets rather than between a vector and a set. Finally note that the nearest neighbor approach is not positive definite (due to the asymmetry of the invariance relation).

We have computed a distance based on our positive definite kernel between every two sets based on underlying kernels of linear and polynomial types. We also computed the nearest neighbor distance between the sets as the minimal distance between points in the sets using the same underlying kernels. The nearest-neighbor approach picked the pair "short comb" and "lines" as the most similar — regardless of the kernel being used (rows (a) and (c) in Fig. 1). The kernel principal angles approach made the same judgment when using the original image space (consistent with the notion that the strength of the approach is based on the ability to work in feature space) but made the correct judgment with the 6'th degree monomial expansion kernel, i.e., determined that rows (a),(b) are the closest pair.

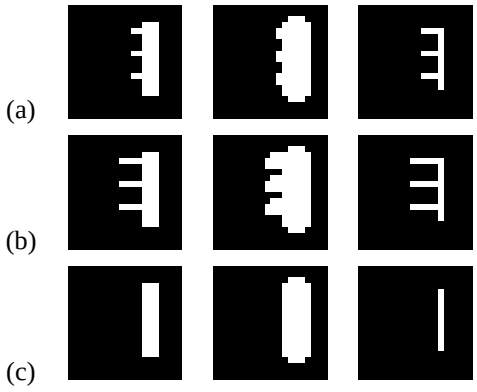

Figure 1: Each row contains an image, the image after a morphological thick operator, and the image after a morphological thin operator. Nearest neighbor approach on this set yields that the set in the first row is closer to the set in the third row then to the set in the second row. Our positive definite kernel, which takes into account all the image space identified that the sets in the first and second row are more similar.

6 Experimental Results

6.1 Detecting Irregular Behavior

Our first experimental example simulates the detection of "irregular" behavior. We note that a behavior is not irregular by itself, but is considered so with respect to some more common "normal" behaviors. Each example in the training and test sets is given as a set of trajectories. Each trajectory is represented by a vector which simulates the location of a person over time. Our goal is to learn to distinguish between homogeneous sets (negative examples) and inhomogeneous sets (positive examples). An inhomogeneous set would contain a trajectory that is different in a sense from the other trajectories in the set. However, the trajectories themselves are not labeled.

Building a real system that would track people over time, and creating a real world test and training sets are outside the scope of our current work. Instead we have simulated the situation using the rules stated below. We define six behavior models. Each behavior model has some freedom with respect to certain parameters. The first model, shown in Figure 2(a) is of straight trajectories. This model, as well as the other more complex models, has the freedom to choose starting and ending points. The next two models, shown in Figures 2(b), 2(c), change their direction once or twice respectively. The exact location of the change can vary slightly as does the orientation of the new direction. The fourth model, shown in Figure 2(d), has a small arc. The starting point of the arc and its length can vary to some extent. The fifth model, shown in Figure 2(e), has a much wider arc, while the last model, shown in Figure 2(f), com-

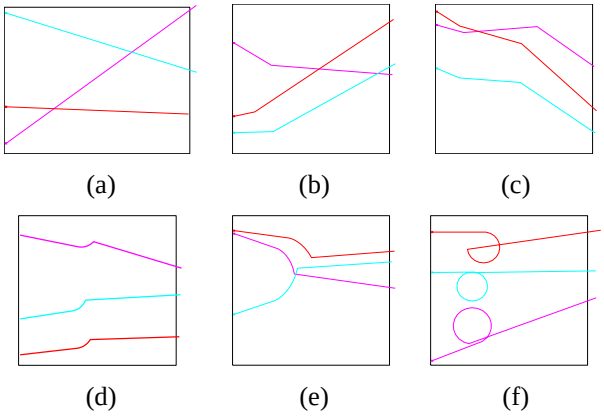

Figure 2: Six models of trajectories. Each figure illustrates some of the variability within one specific model. (a) Straight trajectories. (b) Direction changes once along the trajectories. (c) Direction changes twice. (d) Trajectories with an arc. (e) Trajectories with a wide arc. (f) Trajectories which complete an almost full circle.

pletes almost a full circle before continuing to its original direction. The exact parameters of the circular motion and its starting point can vary.

We used the Support Vector Machine (SVM) [3, 21] algorithm for our classification engine. The SVM was given a training set of input *matrices* $A_1, ..., A_l$ with labels $y_1, ..., y_l$ where $y_i = \pm 1$, where the columns of a matrix A_i represent the trajectories of the i'th "instance" example. The input to the SVM algorithm is a "measurement" matrix M whose entries $M_{ij} = y_i y_j f(A_i, A_j)$ and the output is a set of "support vectors" which consist of the subset of instances which lie on the margin of the positive and negative examples. In our case, the support vectors are matrices. The classification of a new test example A is based on the evaluation of the function:

$$h(A) = sgn(\sum \mu_i y_i f(A, A_i) - b)$$

where the sum is over the support matrices and μ_i are the corresponding Lagrange multipliers provided by the algorithm. Note that it is crucial that our measure $f()$ is a positive definite kernel because otherwise we could not have plugged it in the SVM.

In the first set of experiments we used a different model for each experiment. In each experiment, all trajectories belong to the same single model, but are oriented in one of the following four directions: left to right, right to left, top to bottom and bottom up. Each example contains seven trajectories. A homogeneous set is considered to be a set where all trajectories lie in one direction. An inhomogeneous set is considered to be a set where six trajectories lie in one direction and one trajectory lies in some other direction. We used 400 training examples and 100 test examples for each experiment. The results are shown in Table 1.

Model	$det(Q_A^\top Q_B)^2$	$det(Q_A^\top Q_B)^2$	Vector	Vector
	linear	Deg 6	linear	Deg 6
(a)	F	1%	F	F
(c)	39%	6%	55%	F
(d)	17%	7%	52%	F
(f)	8%	3%	57%	F

Table 1:

The values in the table entries are of error rates for the test set. The experiment was done using our proposed kernel for sets ("$det(Q_A^\top Q_B)^2$") over a linear kernel and over a polynomial kernel of degree 6, and for vector ("Vector") representation of the sets learned using the same kernels.

Each row represents an experiment made using a different model of trajectories. "F" means that the SVM classifier failed to converge. Other kernels suggested in section 4 where also tested but failed to converge or gave very poor results.

The second experiment was similar to the first one, but here we used the first three models together. In this experiment, a homogeneous set includes seven trajectories of the same randomly picked model of the three. All trajectories in a homogeneous set were oriented along the same direction. In an inhomogeneous set, on the other hand, there exists a single trajectory of a different model whose motion was oriented at a random direction which might or might not coincide with the direction of the other six trajectories (see Table 2, first row). We tried a "tougher" variation along the same experimental theme where all the trajectories (regular and irregular) extent from left to right (a single direction). As a result, the irregular trajectory is expressed only by the trajectory model and not by direction (see Table 2, second row).

Directions	$det(Q_A^\top Q_B)^2$	$det(Q_A^\top Q_B)^2$	Vector	Vector
	linear	Deg 6	linear	Deg 6
4	26%	19%	60%	F
1	F	40%	F	F

Table 2:

The third experiment was similar to the second one, only this time we used all six models as possible types of trajectories. Error rates are given in Table 3.

6.2 Face Recognition

In our second experimental example the goal was to recognize a face by matching video sequences. We ran a face tracker on 9 persons who were making head and facial movements in front of a camera. The result of the face

Directions	$det(Q_A^\top Q_B)^2$ linear	$det(Q_A^\top Q_B)^2$ Deg 6	Vector linear	Vector Deg 6
4	27%	15%	47%	F
1	F	35%	F	F

Table 3:

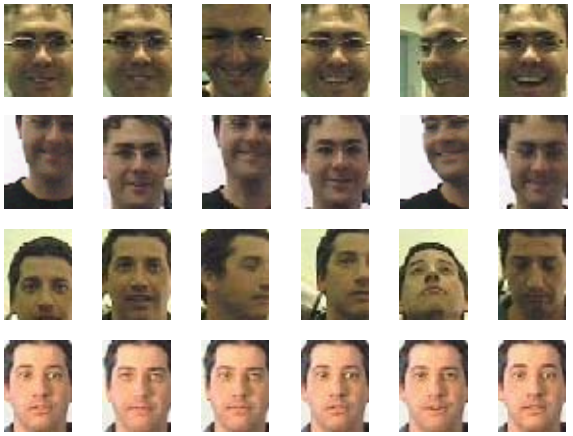

Figure 3: Each pair of rows contains some of the extracted face images of the same person on different shots taken under different illumination. The sequences on the top two rows is a hard example that was not recognized by any method.

tracker is an area of interest bounding the face which was scaled (up or down) to a fixed size of 35×47 per frame per person. The number of images per set varied from 30 to 200 frames per set. Since the tracker is not perfect (none of them are) especially against strong pose changes of the head, some of the elements of the set were not positioned properly on the face (Fig. 3 second row).

The training set consisted of 9 sets (one per person), while the testing set consisted of 7 new sets (of the same people). We performed a matching over sets in order to select the closest two sets between the test set and the 9 training sets. The kernel principal angles was applied once on the original image space and once using a feature space representing the 6'th order monomial expansion of the original input vectors. Since we are not constrained in this experiment to use a positive definite measure, we used the mean of the smallest 20 principal angles as the similarity measure between two video sequences (labeled as "mean θ" in the sequal). Note also that in this kind of experiment, the length of the video sequences can vary.

We compared our results to four other approaches for obtaining a similarity measure over sets. In the second approach (labeled "alt"), instead of computing the principal angles, we chose the angle between the closest vectors in the two sets. At first the two vectors (one from each set) which had the largest inner-product were picked. They were removed and we then picked the next pair and so on. This method is used as a "low cost substitute" for principal angles. The third method (labeled "NN") measured the distance between every two sets as the distance in feature space between their closest elements. Recall that the distance in feature space between two vectors is:

$$d(\phi(\mathbf{x}), \phi(\mathbf{x}'))^2 = k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}, \mathbf{x}').$$

The fourth method (labeled "20NN") examined the 20 vectors in the union of the training sets which were closest to the vectors of the test set. The recognition process was based on a vote - the training set which contributed the most of these vectors was chosen. The last method we compared to was the method based on Kullback-Leibler divergence presented in [18].

One can see from Table 4, that our approach based on computing the principal angles in a feature space of 6'th order monomials made only a single error out of 7 tests (the first two rows of Fig. 3, where as all four other approaches performed poorly.

	Linear	Deg 6
mean θ	2	1
Alt	4	4
NN	5	5
20NN	3	3
[18]	4	NA

Table 4:

7 Summary

In this paper we have made three contributions:

- A case in favor of using instance space over matrices (sets of vectors) for classification. We have shown that the need arises especially in the context of computer vision applications, but not exclusively so as we noted that "kernel jittering" is another case for similarity over sets of vectors.

- A kernel approach for the computation of principal angles in feature space. We noted that principal angles in the original input space is a fairly limited tool for comparing subspaces because of the linearity assumption. However, the linearity assumption in the high-dimensional feature space allows for non-linearities in the input space thereby making the principal angles approach for matrix similarity a powerful tool for matching over sets.

- Introducing the function $f(A, B)$ which forms a positive definite kernel. This result is important for making use of the similarity measure over matrices as a metric for optimal margin classifiers.

Applications of principal angles are found in numerous branches of science including data analysis [11], random processes [14, 10] and stochastic processes (cf. [7]). The power to map the observation onto a high dimensional feature space while working through the process of finding principal angles via simple kernel functions in the original input space would no doubtly become useful beyond the scope of this paper. As for visual understanding applications, the range of examples is not limited to the scope presented here — in fact any visual classification or clustering task applied to non-rigid shapes may benefit from the approach of matching over sets. Finally it is worth noting that the algorithm presented in this paper is general in the sense it holds for any type of sets, i.e., not only sets made out of vectors. Any set for which one can present a kernel function operating on the elements of the set could serve as an input to the algorithm — for some types of sets it is possible to obtain interesting interpretations as to what the algorithm actually does, but that is out of the scope of this paper.

Acknowledgments

The authors thank Adi Shavit, Dana Segal and GenTech corporation for the use of their "head" detection and tracker software.

References

[1] Aitken, *Determinants and Matrices*, Interscience Publishers Inc., 4th edition, 1946.

[2] H. Baird, Document image defect models. In *proceedings, IAPR Workshop on Syntactic and Structural Pattern Recognition*, 1990.

[3] B.E. Boser, I.M. Guyon, and V.N. Vapnik, A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 1992.

[4] Z. Biuk and S. Loncaric. Face Recognition from Multi-Pose Image Sequence. In *Proceedings of 2nd Int'l Symposium on Image and Signal Processing and Analysis*, 2001.

[5] A. Bjork and G.H. Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27(123):579–594, 1973.

[6] R.A Brualdi, S. Friedland, and A. Pothen, The sparse basis problem and multilinear algebra, In *SIAM Journal on Matrix Analysis and Applications, Volume 16*,1995.

[7] P.E. Caines. *Linear Stochastic Systems.* Wiley NY, 1988.

[8] D. DeCoste, and B. Schölkopf. Training invariant support vector machines. In *Machine Learning* 46, 161-190, 2002.

[9] G.J. Edwards, C.J. Taylor, and T.F. Cootes. Improving Identification Performance by Integrating Evidence from Sequences. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1999.

[10] I.M. Gelfand and A.M. Yaglom. Calculation of the amount of information about a random function contained in another such function. *Amer. Math. Soc. Translations, series 2*, 12:199–246, 1959.

[11] R. Gittins. *Cannonical Analysis: A Review with Applications in Ecology.* Biomathematics, Vol. 12, Springer-Verlag, 1985.

[12] S. Gong, A. Psarrou, I. Katsoulis, and P. Palavouzis. Tracking and Recognition of Face Sequences. In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production*, 1994.

[13] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–372, 1936.

[14] T. Kailath. A view of three decades of linear filtering theory. *IEEE trans. on Information Theory*, 20(2):146–181, 1974.

[15] C.S. MacInnes, The Solution to a Structured Matrix Approximation Problem Using Grassman Coordinates In *SIAM Journal on Matrix Analysis and Applications, Volume 21(2)*,1999.

[16] T. Poggio, and T. Vetter. Recognition and structure from one 2D model view: observations on prototypes, object classes and symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[17] B. Schölkopf, and A.J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[18] G. Shakhnarovich, J.W. Fisher, and T. Darrell, Face recognition from long-term observations In *European Conference on Computer Vision (ECCV)*, 2002.

[19] P. Simard, B. Victorri, T. LeCun, and J. Denker. Tangent prop - a formalism for specifying selected invariances in an adaptive network. In *Advances in neural information processing systems 4*, 1992.

[20] M.Turk and A.Pentland. Eigen Faces for Recognition. *Journal of Cognitive Neuroscience*, 3(1), 1991.

[21] V. Vapnik, *Statistical learning theory*. NY:Wiley, 1998.

[22] Osamu Yamaguchi, Kazuhiro Fukui, and Ken-ichi Maeda. Face recognition using temporal image sequence. In *IEEE International Conference on Automatic Face & Gesture Recognition*, 1998.