# Parameterized Kernel Principal Component Analysis: Theory and Applications to Supervised and Unsupervised Image Alignment

Fernando De la Torre Minh Hoai Nguyen.
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA15213.
ftorre@cs.cmu.edu minhhoai@cmu.edu

## Abstract

*Parameterized Appearance Models (PAMs) (e.g. eigentracking, active appearance models, morphable models) use Principal Component Analysis (PCA) to model the shape and appearance of objects in images. Given a new image with an unknown appearance/shape configuration, PAMs can detect and track the object by optimizing the model's parameters that best match the image. While PAMs have numerous advantages for image registration relative to alternative approaches, they suffer from two major limitations: First, PCA cannot model non-linear structure in the data. Second, learning PAMs requires precise manually labeled training data. This paper proposes Parameterized Kernel Principal Component Analysis (PKPCA), an extension of PAMs that uses Kernel PCA (KPCA) for learning a non-linear appearance model invariant to rigid and/or non-rigid deformations. We demonstrate improved performance in supervised and unsupervised image registration, and present a novel application to improve the quality of manual landmarks in faces. In addition, we suggest a clean and effective matrix formulation for PKPCA.*

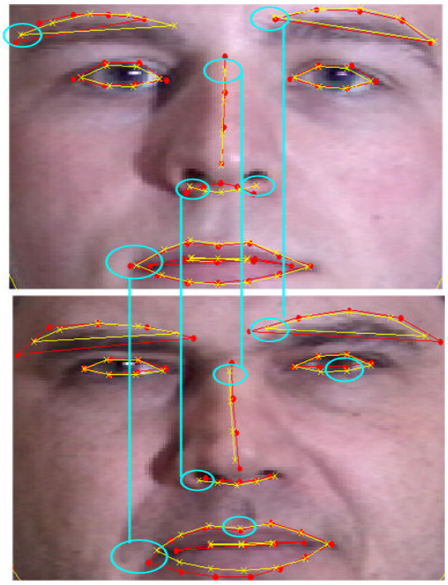

Figure 1. Unsupervised learning of a non-linear generative model of frontal face images. Red dots denote initial manual labeling. Yellow crosses represent the automatic re-labeling using PKPCA. Images are better seen in color.

## 1. Introduction

Since the early work of Sirovich and Kirby [33] parameterizing the human face using Principal Component Analysis (PCA) and the successful eigenfaces of Turk and Pentland [34], many computer vision researchers have used PCA techniques to construct linear models of optical flow, shape or graylevel [5, 6, 8, 26, 21, 7, 16]. The modeling power of PCA techniques is especially useful when applied to visual data, because there is a need for dimensionality reduction given the increase in the number of features.

Parameterized Appearance Models (PAMs) (*e.g.* eigentracking [6], active appearance models [8, 12, 24, 16], morphable models [7, 21, 35]) have proven to be a good statistical tool to build models of shape and appearance variation of objects. In particular, PAMs have been extensively applied to the detection, tracking and synthesis of faces during the last decade. PAMs utilize PCA to model shape and appearance variations across pose, expression, illumination and identity. However, the linear assumption of PCA does not always hold true. For instance, modeling pose changes with 2D models is a very rough approximation (*e.g.* points disappear with occlusion) and non-linear models are preferred. Similarly, the space of frontal faces is likely to be better described by a non-linear model to account for different expressions, illumination, beards or glasses. In fact, numerous papers have shown the advantages of Kernel PCA (KPCA) over PCA in face recognition or image modeling tasks (*e.g.* [38, 28, 37, 25, 30]). This suggests that KPCA is a more powerful model for image analysis, provided that the correct kernel and kernel parameters are known. This paper describes an extension of PAMs, Parameterized Ker-



978-1-4244-2243-2/08/$25.00 ©2008 IEEE

nel PCA (PKPCA), a framework for learning a non-linear generative model of objects' appearance and shape in a supervised and unsupervised manner.

Fig. 1 illustrates an application of PKPCA. The red dots represents the manual labeling done by experienced labelers to build AAMs [8]. The yellow crosses are the result of re-labeling using PKPCA. PKPCA learns, in an unsupervised manner, a non-linear model of frontal faces from 600 images (starting from the red dots). As it can be observed (see blue circles), PKPCA is able to learn a more consistent labeling among subject's faces (*e.g.* nose, corners of the mouth, eyes). Moreover, PKPCA builds a more compact model (less eigenvectors), which is less prone to over-fitting and more computationally efficient.

The rest of this paper is organized as follows. Sec. 2 reviews previous works on image alignment with appearance models. Sec. 3 proposes an energy-based framework for learning KPCA. Sec. 4 derives PKPCA by incorporating rigid and non-rigid transformations into the KPCA formulation. Sec. 5 illustrates the benefits of our approach for rigid/non-rigid supervised and unsupervised image alignment.

## 2. Previous work

Over the last decade, appearance models have become increasingly important in computer graphics and vision. In particular, parameterized appearance models (PAMs) have proven useful in the detection, tracking, and synthesis of human faces from video [7, 6, 5, 14, 8, 24, 11, 26, 21, 16, 35]. One of the main benefits of PAMs is its use of gradient descent methods to align images with high dimensional deformation models.

Although widely used, one of the main limitations of PAMs is its use of a linear model. Several attempts have been made to extend appearance models in order to cope with non-linearities. Cootes *et al.* [10] build view-based Active Appearance Models (AAMs) capable of tracking non-linear pose changes from profile to profile by continuously switching several discrete linear appearance models. However, it remains unclear how to impose consistency of the identity parameters across multiple views. Romdhani *et al.* [28] use KPCA with point distribution models to model the shape out-of-plane motion. However, the KPCA is only applied to the shape, and it is not embedded into the appearance model that drives the AAM fitting. Moreover, finding the pre-image shape is an iterative procedure prone to encountering local minima. More importantly, the model is learned in a supervised manner (i.e. manual labeling). In the context of discriminative models, Avidan [1] has proposed a gradient-based method to make support vector machines invariant to translations.

Supervised image alignment [7, 6, 8, 24, 11] is well understood in the literature. By supervised alignment, we refer to an image that is registered w.r.t. a previously learned model (off-line). Learning the model in an unsupervised manner has been less explored. Frey and Jojic [14] propose a method for learning a factor analysis model invariantly to geometric transformations. The proposed method grows polynomially with the number of possible spatial transformations, and it can be computationally intensive when working with high dimensional motion models. To improve upon this problem, De la Torre and Black [11] propose parameterized component analysis, a gradient-based method that learns a PCA model invariantly to affine transformations. More recently, Miller *et al.* have proposed the Congealing method [23, 19] that uses an entropy measure to align images with respect to the distribution of the data. Baker *et al.* [2] learned an AAM invariantly to rigid and non-rigid motion. Kookinos and Yuille [22] proposed a probabilistic framework and extended previous approaches [2, 23, 11] to deal with articulated objects using a Markov Random Field (MRF) on top of the AAM.

Unlike previous works, we integrate the kernel methods in the core of the AAM framework. In particular, we develop a gradient descent algorithm for the efficiently fitting of kernel appearance models into new images. Moreover, we show how to learn the kernel appearance model in an unsupervised fashion invariantly to rigid and non-rigid transformations. Furthermore, we suggest a clean and effective matrix formulation.

## 3. Energy-based PCA methods

Component Analysis (CA) methods (*e.g.* PCA, LDA, Tensor Factorization) have been successfully applied in numerous classification, clustering and dimensionality reduction tasks in the last two decades. Many CA techniques are especially appealing because they can be formulated as eigen-problems, offering great potential for efficient learning of linear and non-linear data representations without local minima. However, the eigen-formulation often obscures important aspects of the learning process such as understanding normalization factors, reducing effect of noise, dealing with missing data, and learning the kernel. In this section, we review previous work on energy-based functions for PCA using a unified matrix formulation.

### 3.1. Principal component analysis

PCA is a statistical technique useful for dimensionality reduction, see [13, 20] for a review of applications and extensions. Let $\mathbf{D} = [\mathbf{d}_1\ \mathbf{d}_2\ ...\ \mathbf{d}_n]$ be a matrix $\mathbf{D} \in \Re^{d \times n}$, see notation[1], where each column $\mathbf{d}_i$ is a data sample,

[1]Bold capital letters denote a matrix $\mathbf{D}$, bold lower-case letters a column vector $\mathbf{d}$. $\mathbf{d}_j$ represents the $j^{th}$ column of the matrix $\mathbf{D}$. $d_{ij}$ denotes the scalar in the row $i$ and column $j$ of the matrix $\mathbf{D}$ and the scalar $i$-th element of a column vector $\mathbf{d}_j$. All non-bold letters represent scalar vari-

$n$ is the number of training samples, and $d$ is the number of features (pixels). The principal components maximize $\sum_{i=1}^{n} ||\mathbf{B}^T\mathbf{d}_i||_2^2 = ||\mathbf{B}^T\mathbf{\Sigma}\mathbf{B}||_F$, with the constraint $\mathbf{B}^T\mathbf{B} = \mathbf{I}$, and where $\mathbf{\Sigma} = \mathbf{D}\mathbf{D}^T = \sum_i \mathbf{d}_i\mathbf{d}_i^T$ is the covariance matrix (assuming zero mean). The columns of $\mathbf{B} \in \Re^{d\times k}$ (principal components) form an orthonormal basis that spans the principal subspace of the data $\mathbf{D}$. If the effective rank of $\mathbf{D}$ is much less than $d$, we can approximate the column space of $\mathbf{D}$ with $k \ll d$ principal components. The data $\mathbf{d}_i$ can be approximated as a linear combination of the principal components as $\mathbf{d}_i^{rec} = \mathbf{B}\mathbf{B}^T\mathbf{d}_i$ where $\mathbf{c}_i = \mathbf{B}^T\mathbf{d}_i$ are the linear coefficients obtained by projecting the training data onto the principal subspace; that is, $\mathbf{C} = \mathbf{B}^T\mathbf{D} \in \Re^{k\times n}$.

The optimal $\mathbf{B}$ can be computed as the leading eigenvectors of $\mathbf{D}\mathbf{D}^T$ [20]. In cases where $d \gg n$, it will be more convenient to compute the eigenvectors of $\mathbf{D}^T\mathbf{D}$ that are related to the eigenvectors of $\mathbf{D}\mathbf{D}^T$. However, for large data sets of high dimensional data, formulating PCA as an error function [27] and applying numerical optimization algorithms is a more efficient procedure (in both space and time) to compute $\mathbf{B}$. Moreover, error functions provide an easier interpretation and generalization. Several error functions exist of which stationary points are solutions of PCA (i.e. the subspace is the same as PCA). Among them, the most appealing one is formulated as [3, 32]:

$$E_1(\mathbf{B},\mathbf{C}) = \sum_{i=1}^{n} ||\mathbf{d}_i - \mathbf{B}\mathbf{c}_i||_2^2 = ||\mathbf{D} - \mathbf{B}\mathbf{C}||_F \qquad (1)$$

A common approach to optimize eq. 1 alternates between solving for $\mathbf{B}$ while $\mathbf{C}$ is fixed and vice versa. This technique is commonly known as Alternated Least Squares (ALS) or Criss-Cross Regression.

### 3.2. Kernel PCA

KPCA [29, 31] is a popular generalization of PCA that allows non-linear feature extraction. KPCA maps the data to a (usually) higher dimensional space, where the data can be linearly modeled (assuming the correct mapping is found). There is no need to explicitly define the mapping using the "kernel trick". KPCA uses a kernel function that implicitly defines the non-linear mapping.

Consider a lifting of the original points to a higher dimensional space $\mathbf{\Gamma} = [\ \boldsymbol{\phi}(\mathbf{d}_1)\ \boldsymbol{\phi}(\mathbf{d}_2)\ \cdots\ \boldsymbol{\phi}(\mathbf{d}_n)\ ]$ where $\boldsymbol{\phi}$ defines the mapping. The kernelized version of eq. 1 can be written as:

$$E_2(\mathbf{B},\mathbf{C}) = ||\mathbf{\Gamma} - \mathbf{B}\mathbf{C}||_F \qquad (2)$$

ables. *diag* is an operator that transforms a vector to a diagonal matrix or takes the diagonal of the matrix into a vector. $\mathbf{1}_k \in \Re^{k\times 1}$ is a vector of ones. $\mathbf{I}_k \in \Re^{k\times k}$ is the identity matrix. $tr(\mathbf{A}) = \sum_i a_{ii}$ is the trace of the matrix $\mathbf{A}$ and $|\mathbf{A}|$ denotes the determinant. $||\mathbf{A}||_F = tr(\mathbf{A}^T\mathbf{A}) = tr(\mathbf{A}\mathbf{A}^T)$ designates the Frobenius norm of matrix $\mathbf{A}$.

Computing the optimal $\mathbf{B} = \mathbf{\Gamma}\mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}$ and substituting this value into eq. 2, it can be shown that:

$$E_2(\mathbf{C}) \propto -tr(\mathbf{C}\mathbf{K}\mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}) \qquad (3)$$

where $\mathbf{K} = \mathbf{\Gamma}^T\mathbf{\Gamma} \in \Re^{n\times n}$ is the standard kernel matrix. Each element $k_{ij}$ of $\mathbf{K}$ represents the similarity between two samples by means of a kernel function (*e.g.* Gaussian Radial Basis Function, polynomial). Optimizing $E_2$ w.r.t. $\mathbf{C}$ can be achieved by computing the leading eigenvectors of $\mathbf{K}$ ($\mathbf{C}$ is the transpose of the eigenvector matrix). After the diagonalization, $\mathbf{C}\mathbf{K}\mathbf{C}^T = \mathbf{\Lambda}$ and $\mathbf{C}\mathbf{C}^T = \mathbf{I}_k$. The computational cost of the eigen-decomposition is $O(n^3)$ (no sparsity is assumed), where $n$ is the number of samples. In KPCA, it is (usually) not convenient to compute the eigenvectors of $\mathbf{\Gamma}\mathbf{\Gamma}^T$ since the dimension of the matrix can be very high dimensional (including infinity).

For large amounts of data (large $n$), an iterative approach to computing KPCA is computationally more efficient. Recall that $\mathbf{B}$ can be expressed as linear combination of the mapped original data $\mathbf{\Gamma}$. That is, $\mathbf{B} = \mathbf{\Gamma}\boldsymbol{\alpha}$. Substituting this expression into eq. 2 results in:

$$E_3(\boldsymbol{\alpha},\mathbf{C}) = ||\mathbf{\Gamma}(\mathbf{I}_n - \boldsymbol{\alpha}\mathbf{C})||_F \qquad (4)$$

Assuming that $\mathbf{K}$ is invertible, similarly to iterative PCA, we can alternate between computing $\boldsymbol{\alpha}$ and $\mathbf{C}$ as:

$$\boldsymbol{\alpha} = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}\ \&\ \mathbf{C} = (\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha})^{-1}\boldsymbol{\alpha}^T\mathbf{K} \qquad (5)$$

The computational cost of each iteration is $O(n^2k)$.

## 4. Parameterized KPCA

Previous section has related the KPCA learning problem to an alternate optimization and made use of the "kernel trick" for effective optimization. This section demonstrates how to parameterize KPCA to compensate for rigid and non-rigid motion. In particular, we show how to register a new image w.r.t. a previously learned KPCA model (supervised), and how to learn the KPCA model invariantly to non-rigid geometric transformations (unsupervised).

### 4.1. Supervised registration with PKPCA

This section extends previous work on Eigentracking [6], AAMs [8] and morphable models [7, 21]) by incorporating KPCA into the formulation.

#### 4.1.1 Rigid motion

We parameterize the image $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{a})) \in \Re^{d\times 1}$ with a rigid geometric transformation $\mathbf{f}(\mathbf{x},\mathbf{a})$ [6, 8, 21]. In the case of an affine transformation, $\mathbf{f}(\mathbf{x},\mathbf{a}) = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} + \begin{pmatrix} a_3 & a_4 \\ a_5 & a_6 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$ where $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5, a_6)$ are

the affine parameters and $\mathbf{x} = (x_1, y_1, \cdots, x_n, y_n)$ is a vector containing the coordinates of the pixels of a given image region. Once the image has been parameterized, the supervised alignment problem can be defined as recovering the motion parameters $\mathbf{a}$ that align the image w.r.t the kernel subspace, that is, minimizing:

$$E_4(\mathbf{c}_n, \mathbf{a}) = ||\boldsymbol{\phi}\big(\mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a}))\big) - \mathbf{B}\mathbf{c}_n||_2^2 \tag{6}$$

Using the fact that $\mathbf{B}^T\mathbf{B} = \boldsymbol{\Lambda}$, and defining $k(\mathbf{x}, \mathbf{y})$ as the kernel function, $E_4$ can be rewritten as:

$$\begin{aligned} E_4(\mathbf{a}) =& k(\mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a}), \mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a}))- \\ & \mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a}))^T\boldsymbol{\Gamma}\mathbf{C}\boldsymbol{\Lambda}^{-1}\mathbf{C}^T\boldsymbol{\Gamma}^T\mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a})) \end{aligned} \tag{7}$$

In eq. 7, we have marginalized the parameter $\mathbf{c}_n$ from the optimization process. However, the minimization problem remains highly non-linear. Recall that $\mathbf{C}$ is the matrix containing the eigenvectors of the kernel matrix, $\mathbf{K}$, learned in the training process. To optimize over the motion parameters, $\mathbf{a}$, we use a Gauss-Newton [4, 6] descent scheme with closed-form increments as in [11]. Following previous work in optical flow and appearance tracking [4, 6, 8, 11], we expand the image changes using Taylor series. $\mathbf{a}^0$ is the initial motion estimation of rigid parameters and $\boldsymbol{\Delta}\mathbf{a}$ is the motion increment.

$$\mathbf{d}\big(\mathbf{f}(\mathbf{x}, \mathbf{a}^0 + \boldsymbol{\Delta}\mathbf{a})\big) = \mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a}^0)) + \mathbf{J}_\mathbf{a}(\mathbf{a}^0)\boldsymbol{\Delta}\mathbf{a} + h.o.t. \tag{8}$$

$h.o.t.$ denotes higher order terms of the expansion. $\mathbf{J}_\mathbf{a}(\mathbf{a}^0) = [\frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a}^0))}{\partial a_1} \quad \cdots \quad \frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a}^0))}{\partial a_6}]$ is the Jacobian matrix evaluated at $\mathbf{a}^0$. To optimize over $\mathbf{a}$ in the case of the RBF Gauassian kernel, we use a fixed-point updating. $E_4$ can be rewritten as:

$$\begin{aligned} & E_4(\mathbf{a}) \propto -\mathbf{r}^T\mathbf{M}\,\mathbf{r} \\ & \text{with } \mathbf{M} = \mathbf{C}\boldsymbol{\Lambda}^{-1}\mathbf{C}^T \;\&\; r_i = e^{-\gamma||\mathbf{p}_i + \mathbf{J}\boldsymbol{\Delta}\mathbf{a}||_2^2} \;\forall i \end{aligned} \tag{9}$$

After differentiating $E_4$ w.r.t. $\boldsymbol{\Delta}\mathbf{a}$ and setting it to zero, eq. 7 can be updated as:

$$\begin{aligned} \frac{\partial E_4}{\partial \boldsymbol{\Delta}\mathbf{a}} &= \sum_{ij} w_{ij}(4\mathbf{J}^T\mathbf{J}\boldsymbol{\Delta}\mathbf{a} + 2\mathbf{J}^T\mathbf{p}_i + 2\mathbf{J}^T\mathbf{p}_j) = 0 \\ \mathbf{p}_i &= \mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{a}^0)) - \mathbf{d}_i, \quad w_{ij} \approx m_{ij}e^{-\gamma||\mathbf{p}_i||_2^2 - \gamma||\mathbf{p}_j||_2^2} \\ \boldsymbol{\Delta}\mathbf{a} &= -(\sum_{ij} 2w_{ij}\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\sum_{ij} w_{ij}(\mathbf{p}_i + \mathbf{p}_j) \\ &= -\frac{1}{\mathbf{1}_n^T\mathbf{W}\mathbf{1}_n}(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{P}\mathbf{W}\mathbf{1}_n \end{aligned} \tag{10}$$

#### 4.1.2 Non-rigid motion

In the previous section, we have parameterized the data with a rigid transformation. In many situations, however, it is interesting to recover non-rigid motion (*e.g.* modeling facial expression). In this section, we propose an extension of section 4.1.1 that takes into account non-rigid motion.

A simple way to incorporate non-rigid motion is to change the definition of $\mathbf{f}$. Consider $\mathbf{f}(\mathbf{B}^s\mathbf{c}^s, \mathbf{a}) = \mathbf{f}(\sum_{i=1}^k c_i^s\mathbf{b}_i^S, \mathbf{a})$, where $\mathbf{B}^s$ is a non-rigid shape model learned by computing PCA on a set of registered shapes [9]. $\mathbf{c}^s$ represent the non-rigid parameters, and $\mathbf{a}$ denote the rigid parameters. In this case, $\mathbf{f}(\mathbf{B}^s\mathbf{c}^s, \mathbf{a})$ will model rigid and non-rigid motion. Aligning a new image w.r.t. the non-rigid model is done minimizing:

$$\begin{aligned} E_5(\mathbf{a}, \mathbf{c}_n, \mathbf{c}_s) &= ||\boldsymbol{\phi}\big(\mathbf{d}(\mathbf{f}(\mathbf{B}^s\mathbf{c}_s, \mathbf{a}))\big) - \mathbf{B}\mathbf{c}_n||_2^2 \\ &= k(\mathbf{d}(\mathbf{f}(\mathbf{B}^s\mathbf{c}_s, \mathbf{a}), \mathbf{d}(\mathbf{f}(\mathbf{B}^s\mathbf{c}_s, \mathbf{a}))- \\ & \quad \mathbf{d}(\mathbf{f}(\mathbf{B}^s\mathbf{c}_s, \mathbf{a}))^T\boldsymbol{\Gamma}\mathbf{C}\boldsymbol{\Lambda}^{-1}\mathbf{C}^T\boldsymbol{\Gamma}^T\mathbf{d}(\mathbf{f}(\mathbf{B}^s\mathbf{c}_s, \mathbf{a}))) \end{aligned} \tag{11}$$

Similar to the rigid case, we make a first order approximation of: $\mathbf{d}\big(\mathbf{f}(\mathbf{B}^s(\mathbf{c}_s + \boldsymbol{\Delta}\mathbf{c}_s), \mathbf{a} + \boldsymbol{\Delta}\mathbf{a})\big) = \mathbf{d}(\mathbf{f}(\mathbf{B}^s\mathbf{c}_s^0, \mathbf{a}^0)) + \mathbf{J}_\mathbf{s}(\mathbf{c}_s^0, \mathbf{a}^0)\boldsymbol{\Delta}\mathbf{s} + h.o.t.$, where $\mathbf{s} = [\mathbf{a}\ \mathbf{c}_s]$. The updates are equivalent to the rigid motion case of eq. (10); the difference is the use of a different Jacobian $\mathbf{J}_\mathbf{s} = \frac{\partial \mathbf{d}\left(\mathbf{f}(\mathbf{B}^s\mathbf{c}_s^0, \mathbf{a}^0)\right)}{\partial \mathbf{s}}$. We omit the expressions in the interest of space.

### 4.2. Unsupervised registration with KPCA

In Sec 3.2, the KPCA has been learned off-line from a set of manually labeled or segmented images. The KPCA was used in section 4.1 for supervised alignment. However, manually labeling images is often time consuming and error prone. This section extends previous expressions (Eq. 7 and Eq. 6) to learn the KPCA model in an unsupervised manner.

Learning KPCA invariantly to rigid and non-rigid geometric transformations requires learning $\mathbf{B}$ and $\mathbf{B}^s$. The unsupervised alignment problem minimizes:

$$\begin{aligned} E_6(\mathbf{A}, \mathbf{C}^a, \mathbf{C}^s, \mathbf{B}, \mathbf{B}^s) &= \sum_{i=1}^n ||\boldsymbol{\phi}\big(\mathbf{d}_i(\mathbf{f}(\mathbf{B}^s\mathbf{c}_i^s, \mathbf{a}_i))\big) - \mathbf{B}\mathbf{c}_i^a||_2^2 \\ \text{subject to} \quad & \mathbf{B}^T\mathbf{B} = \boldsymbol{\Lambda}, \;\&\; \mathbf{B}^{sT}\mathbf{B}^s = \mathbf{I}_s \end{aligned} \tag{12}$$

w.r.t. to the rigid motion parameters $\mathbf{A} = [\mathbf{a}_1 \ \cdots \ \mathbf{a}_n]$, the appearance coefficients $\mathbf{C}^a = [\mathbf{c}_1^a \ \cdots \ \mathbf{c}_\mathbf{n}^\mathbf{a}]$, the shape coefficients $\mathbf{C}^s$, the shape bases $\mathbf{B}^s$, and appearance bases $\mathbf{B}$. The algorithm alternates between two steps: the first step, registers each of the images w.r.t. to an initial model by computing $\mathbf{A}$ and $\mathbf{C}^s$, while $\mathbf{C}^a$ is marginalized. The second step recomputes the matrix $\mathbf{C}^a$ (eigenvectors of $\mathbf{K}$) and $\mathbf{B}^s$ using the new aligned landmarks. The $\mathbf{B}^s$ matrix contains the modes of shape variation that preserve $x\%$ of shape energy (typically $90\%$), after performing procrustes in the shape landmarks. After the first iteration, we add additional modes in $\mathbf{B}^s$ to allow translational movement of some of the landmarks, otherwise $\mathbf{B}^s$ would be the same at each iteration.

## 5. Experiments

In this section, we report experimental results for supervised and unsupervised image registration, and compare the

results to previous methods for rigid appearance registration (Eigentracking [6]) , non-rigid registration (AAMs [8]), and unsupervised registration (Congealing [23]).

## 5.1. Supervised alignment

This section highlights the benefits of registering with KPCA rather than linear PCA for rigid (i.e. [6]) and rigid/non-rigid (i.e. [8]) motion models.

### 5.1.1 Rigid appearance registration

Many recognition algorithms operate on the basis that the object to be recognized is seen in a canonical pose. In this experiment, we show how to register an object with respect to a generic model that contains all possible classes. Since the classes are very diverse, it is unlikely that a linear assumption will hold.

We selected 48 objects from the Amsterdam Library of Object Images database [15]. Each object is recorded under 8 different illumination conditions and the image size is 72×96. Some examples of objects are given in fig. 2.

Figure 2. Some images from the ALOI database

For each object, seven images were selected for training and the last one was used for testing. We shifted the test image five pixels horizontally and five pixels vertically. Using an affine transformation as rigid motion, we tried to recover the translation using Eigentracking (PCA) [6] and supervised PKPCA (rigid). We retained 70% of the energy for both methods, which was the best setting for both. Fig. 3 plots the errors for each of the 48 images. The error is the difference between the recovered translation and the initial perturbation, *i.e.* $|t_x - 5| + |t_y - 5|$. As shown in fig. 3, PKPCA allows for more accurate registration starting from the same initial configuration. At first, this result seems to be counter intuitive because images of the same object at different illuminations often form a linear subspace (assuming Lambertian surfaces). However two noteworthy factors exist: First, the linearity assumption breaks down when combining images of different objects. Second, the object surfaces are not Lambertian and there are shadow effects. In this experiment, we have used the RBF Gaussian kernel for KPCA. Fig. 3 shows the reconstruction error, PKPCA achieves better reconstruction and alignment.

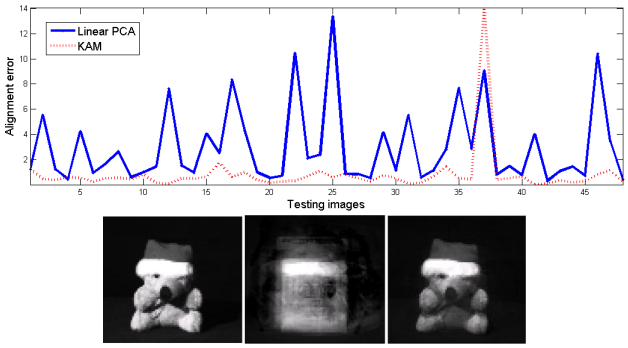


Figure 3. Top: error versus testing images. Bottom: left) original image, center) PCA reconstruction, right) KPCA reconstruction

### 5.1.2 Registering faces with PKPCA

Accurate alignment of faces is a very important step in applications such as facial expression analysis or face recognition. AAMs and MMs [8, 12, 16, 24, 35] are state-of-the-art algorithms for face alignment. In this section, we compare PKPCA and AAMs for non-rigid alignment on the CMU Multi-PIE database [18].

We randomly select 700 face images (120×160) containing 5 different expressions: smile, disgust, surprise, scream and squint (roughly 140 images each). All images are frontal and are taken under the same illumination conditions. Each face is manually labeled with 68 landmarks as shown in fig. 4a. A PCA model of shape is built retaining 80% of the energy [8, 12, 24]. The total number of parameters to recover is 12, six for affine transformation and another six for shape variation. For appearance modeling, we extract the intensity values of pixels inside the rectangular patches around the landmarks as in [12]. Figure 4b shows an example of patches for landmarks around the mouth area. Both PKPCA and linear PCA alignment systems are trained by retaining 80% of the energy. We use 100 testing images

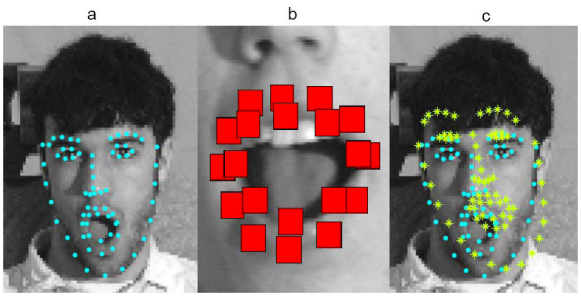


Figure 4. (a) example of landmarks associated with each face, (b) patches for appearance modeling, (c) example of shape distortion

and randomly perturb the affine and non-rigid transformation with increasing power. Figure 4.c shows an example of such perturbation. The correct landmarks are marked in cyan (circles), while the transformed shape is shown in yel-

low (stars). Note that none of the subjects in the testing images are in the training set. For each testing image, we record the sum of absolute differences between the ground truth landmarks and the recovered ones. Fig. 5 shows the average and standard deviation of the alignment error as a function of the amount of perturbation. As can be observed, PKPCA provides better alignment in comparison with linear PCA, for the same percentage of energy preserved in the models. The difference is especially significant for large amounts of perturbation.

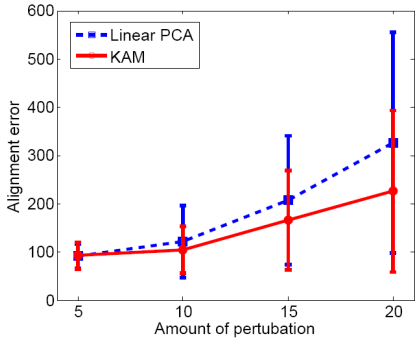


Figure 5. Aligment error versus amount of perturbation.

## 5.2. Unsupervised alignment

This section shows the benefits of unsupervised alignment in two data sets, the USPS dataset and Multi-PIE [18].

### 5.2.1 USPS data set

Fig. 6 shows the results of jointly registering and learning the model for the USPS data set. We randomly select a set $\mathbf{S}$ of 100 images ($16\times16$) of ten hand-written digits from the USPS dataset (fig. 6.a). Figure 6.b shows $\mathbf{S}$ after 5 iterations. The energy amounts preserved by KPCA at iterations 1, 2, 3, 4 and 5 are 70%, 72.5%, 75%, 77.5%, and $80\%$ respectively. The initial set $\mathbf{S}$ requires 33 principal components to preserve 80% of the energy. After jointly aligning the data and computing PKPCA, only 6 eigen-bases are needed to preserve $80\%$ of the energy. This indicates that after convergence, we obtain a more compact model, likely to have better generalization properties.

For comparison purposes, we also perform unsupervised alignment using the Congealing method [23]. To avoid warping outside image boundaries, we pad the images with 15-pixel white borders on each side. The experiment is done using the code provided by the author [2] for affine transformation with 80 iterations. The result is given in Fig. 6c. As can be seen, our method produces better visual results. Moreover, the number of bases to encode data after alignment (using PCA with $80\%$ energy) is 14, twice as many as our method (6). In the experiments reported by [23], the authors jointly aligned samples of the same number, achieving better alignment results than the ones in fig. 6.c. Furthermore, our method is more computationally efficient. Employing a Pentium 4 3Ghz running Windows XP our method takes 25s, while the Congealing method takes 305s.

[2] $http://www.cs.umass.edu/\sim elm/congealing/$

## 5.3. Improving face labeling

This section shows an application of PKPCA to improve upon manual labeling of landmarks in faces. Previous work of Walker *et al.* [36] has addressed this important problem by finding correspondences among salient points in images. Recently [19] proposed an extension of Congealing methods to align faces with rigid transformations. In this section, we demonstrate how to extend previous work in order to align facial features of people's faces using PKPCA.

We selected 920 frontal face images with neutral expression and direct illumination (see fig. 1) from the CMU Multi-PIE database [18]. We use 600 face images to learn a KPCA model of appearance, $\mathbf{C}$ (eq. 3), around 66 landmarks and a linear model of shape, $\mathbf{B}^s$. We perform PCA on the shape, since it has been previously shown that a linear model provides good generalization across identities and expressions; whereas, a linear model does not generalize well for appearance [17]. The algorithm starts with the manually labeled landmarks red dots in fig.(7). Different images have been labeled by different people. $\mathbf{B}^s$ is obtained by computing PCA on the landmarks and preserving $95\%$ of the energy. An a additional translational basis ($x$ and $y$ directions) is added for the landmarks in the eyes, brows and corners of the mouth. These extra basis will allow compensation for more accurate positioning of the landmarks outside the shape model. The KPCA model ($\mathbf{C}$) is computed by performing KPCA on patches of $22 \times 22$ pixels around the landmark locations [12], and preserving $90\%$ of the energy. Once the initial model is built, the algorithm alternates between two steps until convergence: (i) align all images w.r.t. the KPCA model, (ii) Recompute the KPCA model so it minimizes Eq. 12.

We let the algorithm run for 10 iterations. At each iteration the number of eigenvectors decreases, since the data is better aligned, and hence more compactly encoded. The amount of eigenvectors for those ten iterations is: $139, 106, 97, 93, 91, 89, 88, 87, 86, 85, 85$. As we can see at the end of the convergence, the KPCA is more compact ($41\%$ less eigenvectors that the initial configuration). Furthermore, the landmarks are placed more consistently across people's features. Fig. 7 shows some examples of landmarks before and after learning KPCA. The yellow crosses correspond to the placement of the landmarks after learning the model invariantly to non-rigid deformations. The red dots represent the initial manual labeling. As we can see, PKPCA is able to achieve a more consistent label-

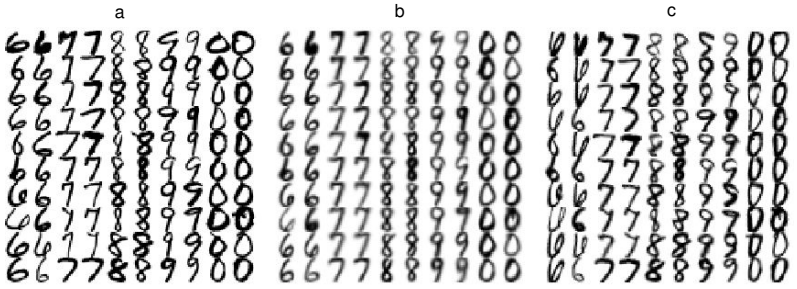


Figure 6. Joint alignment of handwritten digits, (a) original digits from USPS dataset, (b) result of PKPCA, (c) result of Congealing method.

ing across subjects' features (*e.g.* nose, eyes).

Since the PKPCA model has fewer parameters, it is likely to be less prone to over-fitting in new testing data. To test the fitting capabilities of the new model, we took $320$ testing subjects from the CMU Multi-pie database [18]. We start from the mean shape and let the PKPCA algorithm converge. The error and standard deviation for the initial model is $3.73 \pm 3.8$ pixels and PKPCA $3.42 \pm 3.1$. The difference is not statistically significant in this dataset, but recall that the PKPCA model uses $41\%$ fewer number of eigenvectors.

## 6. Conclusion

In this paper, we have extended KPCA by incorporating geometric transformations into the formulation, and a gradient descent algorithm has been proposed for fast alignment. Furthermore, we show how to learn this model in an unsupervised manner. Preliminary experiments show the benefits of our approach in comparison with traditional linear PCA models to register both rigid and non-rigid motion.

**Acknowledgments** This work was partially supported by National Institute of Justice award 2005-IJ-CX-K067 and National Institute of Health Grant R01 MH 051435.

## References

[1] S. Avidan. Support vector tracking. In *Conference on Computer Vision and Pattern Recognition*, 2001.

[2] S. Baker, I. Matthews, and J. Schneider. Automatic construction of active appearance models as an image coding problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1380 – 1384, October 2004.

[3] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

[4] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. *European Conference on Computer Vision*, 1992.

[5] M. J. Black, D. J. Fleet, and Y. Yacoob. Robustly estimating changes in image appearance. *Computer Vision and Image Understanding*, 78(1):8–31, 2000.

[6] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of objects using view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[7] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999.

[8] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *PAMI*, 23(6):681–685, 2001.

[9] T. Cootes and C. Taylor. Statistical models of appearance for computer vision. In *Tech. report, Univ. of Manchester*, 2001.

[10] T. Cootes, K. Walker, and C. Taylor. View-based active appearance models. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[11] F. de la Torre and M. J. Black. Robust parameterized component analysis: theory and applications to 2d facial appearance models. *Computer Vision and Image Understanding*, 91:53 – 71, 2003.

[12] F. de la Torre, A. Collet, J. Cohn, and T. Kanade. Filtered component analysis to increase robustness to local minima in appearance models. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[13] K. I. Diamantaras. *Principal Component Neural Networks (Therory and Applications)*. John Wiley & Sons, 1996.

[14] B. J. Frey and N. Jojic. Transformation-invariant clustering and dimensionality reduction. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, to appear.

[15] J. Geusebroek, G. Burghouts, and A. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[16] S. Gong, S. Mckenna, and A. Psarrou. *Dynamic Vision: From Images to Face Recognition*. Imperial College Press, 2000.

[17] R. Gross, I. Matthews, and S. Baker. Generic vs. person specific active appearance models. *Image and Vision Computing*, 23(11):1080–1093, 2005.

[18] R. Gross, I. Matthews, J. F. Cohn, T. Kanade, and S. Baker. The CMU Multi-Pose, Illumination, and Expression (Multi-PIE) face database. Technical report, Carnegie Mellon University Robotics Institute.TR-07-08, 2007.

Figure 7. Red dots represent the initial manual labeling. Yellow crosses are the landmarks re-labeled with PKPCA (better seen in color).


[19] G. Huang, V. Jain, and E. Learned-Miller. Unsupervised joint alignment of complex images. In *International Conference on Computer Vision*, October, Brasil 2007.

[20] I. T. Jolliffe. *Principal Component Analysis*. New York: Springer-Verlag, 1986.

[21] M. J. Jones and T. Poggio. Multidimensional morphable models. In *International Conference on Computer Vision*, pages 683–688, 1998.

[22] I. Kookinos and A. Yuille. Unsupervised learning of object deformation models. In *International Conference on Computer Vision*, 2007.

[23] E. G. Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, 2006.

[24] I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164, Nov. 2004.

[25] T. Melzer, M. Reiter, and H.Bischof. Kernel cca: A nonlinear extension of canonical correlation analysis. *International Conference on Artificial Neural Networks*, 2001.

[26] S. K. Nayar and T. Poggio. *Early Visual Learning*. Oxford University Press, 1996.

[27] E. Oja. A simplified neuron model as principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[28] S. Romdhani, S. Gong, and A. Psarrou. A multiview nonlinear active shape model using kernel pca. In *British Machine Vision Conference*, 1999.

[29] B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[30] B. Schölkopf, A. Smola, and K. Mller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[31] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[32] H. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *Pattern Analysis and Machine Intelligence*, 17(9):855–867, 1995.

[33] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4(3):519–524, March 1987.

[34] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal Cognitive Neuroscience*, 3(1):71–86, 1991.

[35] T. Vetter. Learning novel views to a single face image. In *Automatic Face and Gesture Recognition*, 1997.

[36] K. Walker, T. Cootes, and C. Taylor. Automatically building appearance models from image sequences using salient features. *Image and Vision Computing*, 20(5-6):435–440, 2002.

[37] S. G. Y. Li and H. Liddell. Recognising trajectories of facial identities using kernel discriminant analysis. *Image and Vision Computing*, 21(13-14):1077–1086, 2003.

[38] M. Yang, N. Ahuja, and D. Kriegman. Face recognition using kernel eigenfaces. In *International Conference on Image Processing*, 2003.