# Pose Invariant Face Recognition Using Linear Pose Transformation in Feature Space

Hyung-Soo Lee and Daijin Kim

Pohang University of Science and Technology
Department of Computer Science and Engineering
San 31, Hyoja-Dong, Nam-Gu, Pohang, 790-784, Korea
{sooz,dkim}@postech.ac.kr

**Abstract.** Recognizing human face is one of the most important part in biometrics. However, drastic change of facial pose makes it a difficult problem. In this paper, we propose linear pose transformation method in feature space. At first, we extracted features from input face image at each pose. Then, we used extracted features to transform an input pose image into its corresponding frontal pose image. The experimental results show that recognition rate with pose transformation is much better than the result without pose transformation.

## 1 Introduction

In modern life, the need for personal security and access control becomes important issue. Biometrics is a technology which is expected to replace traditional authentication methods which is easy to be stolen, forgotten and duplicated. Fingerprints, face, iris, and voiceprints are commonly used biometric features. Among these features, face provides more direct, friendly and convenient identification way and is more acceptable compared with individual identification ways of other biometrics features[1]. Thus, face recognition takes one of the most important parts in biometrics. Many researchers have investigated face recognition. However, face appearance in nature scenes varies drastically with changes of facial pose, illumination conditions and so forth. Such variations make face recognition process difficult. Among these difficulties, pose variation is the most critical and challenging one.

Beymer[2], Biuk[3],and Huang[4] divided face images into several subsets according to facial angles and model each view subspace respectively. Then they estimated the pose angle of input facial image and projected the image onto the corresponding subspace. Finally they classified the face image in projected subspace. Such view-based scheme is preferred because it is avoided to explicitly establish 3D model from each pose image, which often tends to be a more complicate problem.

In this paper, we propose linear pose transformation method in feature space. First we compute subspace of each pose images using PCA or kernel PCA. Then we compute pose transformation matrix between input pose subspace and frontal

N. Sebe et al. (Eds.): HCI/ECCV 2004, LNCS 3058, pp. 211–220, 2004.

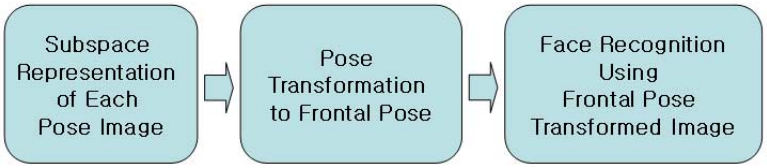


**Fig. 1.** Process of Pose Transformation

pose subspace. We can represent any input face by a linear combination of basis vectors. If we have a pair of posed facial image and its corresponding frontal facial image for the same person, we can obtain an estimation of the transformation between posed image and frontal image by using the coefficients of training data. Using obtained pose transformation matrix, we can transform any input posed image to frontal posed image. Finally we have images transformed to frontal pose, then we can use usual face recognition method such as LDA[5], GDA[6], NDA[7][8] and nearest neighbor. Fig.1 shows the process of our proposed method.

## 2 Subspace Representation

Input facial image is generally very high-dimensional data and pose transformation in input space shows usually low performance. Therefore we need to represent facial image in subspace not only for dimensionality reduction, but also for relevant feature extraction. In this section, we review two methods of subspace representation PCA and kernel PCA.

### 2.1 Principal Component Analysis

From the viewpoint of both the curse of dimensionality and the optimality of the pattern classification, it is desirable to reduce the dimensionality of feature space of the data. In PCA[9], a set of observed $n$-dimensional data vector $\boldsymbol{X} = \{\boldsymbol{x}_p\}$, $p \in \{1, \cdots, N\}$ is reduced to a set of $m$-dimensional feature vector $\boldsymbol{S} = \{\boldsymbol{s}_p\}$, $p \in \{1, \cdots, N\}$ by a transformation matrix $T$ as

$$\boldsymbol{s}_p = T^t(\boldsymbol{x}_p - \mathcal{E}[\boldsymbol{x}]), \tag{1}$$

where $m \leq n$, $T = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m)$ and the vector $\boldsymbol{w}_j$ is the eigenvector which corresponds to the $j$th largest eigenvalue of the sample covariance matrix $C = \frac{1}{N}\sum_{p=1}^{N}(\boldsymbol{x}_p - \mathcal{E}[\boldsymbol{x}])(\boldsymbol{x}_p - \mathcal{E}[\boldsymbol{x}])^T$, such that $C\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k$. The $m$ principal axes $T$ are orthonormal axes onto which the retained variance under projection is maximal. One property of PCA is that projection onto the principal subspace minimizes the squared reconstruction error $\sum \parallel \boldsymbol{x}_p - \hat{\boldsymbol{x}} \parallel^2$. The optimal linear reconstruction of $\hat{\boldsymbol{x}}$ is given by $\hat{\boldsymbol{x}} = T\boldsymbol{s}_p + \mathcal{E}[\boldsymbol{x}]$, where $\boldsymbol{s}_p = T^t(\boldsymbol{x}_t - \mathcal{E}[\boldsymbol{x}])$, and the orthogonal columns of $T$ span the space of the principal $m$ eigenvectors of $C$.

### 2.2 Kernel Principal Component Analysis

Kernel PCA[10] computes the principal components in a high-dimensional feature space, which is nonlinearly related to the input space. The basic idea of kernel PCA is that first map the input data $x$ into a high-dimensional feature space $F$ via a nonlinear mapping $\Phi$ and then perform a linear PCA in $F$. We assume that we are dealing with centered data,i.e., $\sum_{i=1}^{N} \Phi(x_i) = 0$, where N is the number of input data. Kernel PCA diagonalizes the covariance matrix of the mapped data $\Phi(x_i)$

$$C = \frac{1}{N} \sum_{i=1}^{N} \Phi(x_i) \cdot \Phi(x_i). \tag{2}$$

To do this, one has to solve the eigenvalue equation $\lambda v = Cv$ for eigenvalues $\lambda \geq 0$ and $v \in F \setminus \{0\}$. As $Cv = \frac{1}{N} \sum_{i=1}^{N} (\Phi(x_i) \cdot v)\Phi(x_i)$, all solutions v with $\lambda \neq 0$ lie within the span of $\Phi(x_1), \cdots \Phi(x_N)$. Thus there exists coefficients $\alpha_i (i = 1, \ldots, N)$ such that

$$v = \sum_{i=1}^{N} \alpha_i \Phi(x_i) \tag{3}$$

If we consider the following set of equations,

$$\lambda(\Phi(x_i) \cdot v) = (\Phi(x_i) \cdot Cv) \text{ for all } i = 1, \ldots, N. \tag{4}$$

we can substitute (2) and (3) into (4). By defining an $N \times N$ matrix K by $K_{ij} \equiv (\Phi(x_i) \cdot \Phi(x_j))$, we arrive at a problem which is cast in terms of dot products: solve

$$N\lambda\alpha = K\alpha \tag{5}$$

where $\alpha = (\alpha_1, \ldots, \alpha_N)^T$. Normalizing the solutions $v^k$, i.e.$(v^k \cdot v^k) = 1$ translates into $\lambda_k(\alpha^k \cdot \alpha^k) = 1$. To extract nonlinear principal components of a test data x, we compute the projection onto the k-th component by $\beta_k := (v^k \cdot \Phi(x)) = \sum_{i=1}^{N} \alpha_i^k k(x, x_i)$.

## 3 Pose Transformation

Our goal is to generate a frontal pose image of unseen test facial image, given its image at certain pose. At first, we extracted features from input face image at each pose using PCA and kernel PCA. Then, we use extracted features to transform an input pose image into its corresponding frontal pose image, i.e. we compute the relation between features at the input pose image and the features at the corresponding frontal pose. A given facial image at each pose can be represented as linear combination of basis vectors at that pose as shown in Fig.2. If we could have the relationship between the coefficient of input pose and that of frontal pose, we can also get the frontal image of input image, because we know each basis vectors of input pose and frontal pose from the training phase. This is our main idea of pose transformation between a given pose and a frontal pose.

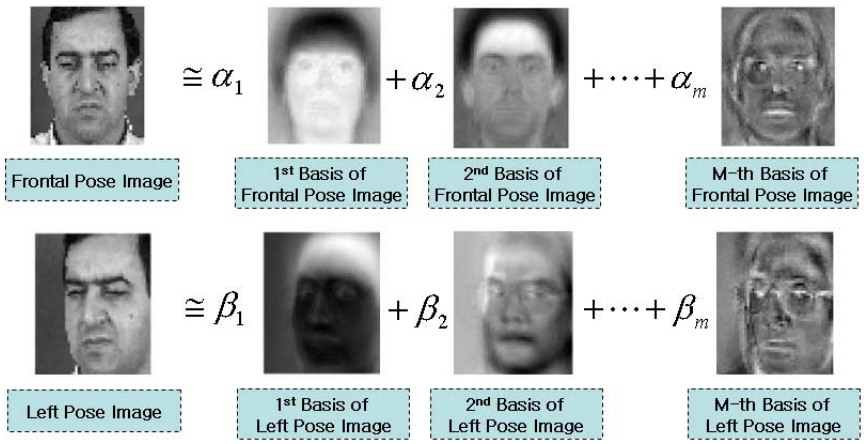


**Fig. 2.** Representing Input Image by Linear Combination of Basis Vectors

### 3.1 Obtaining Transformation Matrix by Least Square Estimation

We use m basis functions, $\Phi^F$ and $\Phi^P$ for subspace representation for each pose image, with $m \leq N$, the number of training images at each pose. From the training set covariance matrix, the $\Phi^F$ and $\Phi^P$ are known and for a given image at pose P and corresponding frontal image are represented as Fig.2. If we define linear pose transformation matrix between pose P and pose F, $\boldsymbol{U} = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_m]$, we can derive following equation :

$$\alpha_i = \boldsymbol{u}_i^T \boldsymbol{\beta} \tag{6}$$

$$= \boldsymbol{\beta}^T \boldsymbol{u_i} \tag{7}$$

where $\alpha_i$ is $i$-th coefficient of frontal image, $\boldsymbol{u}_i = (u_{i,1} \cdots u_{i,m})$ is $i$-th vector of transformation matrix $\boldsymbol{U}$, and $\boldsymbol{\beta} = (\beta_1 \cdots \beta_m)$ is coefficient vector of pose image. We have $N$ training pair images, so we can use least square estimation to solve equation 7. Equation 7 is equivalent to the following equation.

$$\begin{pmatrix} \alpha_i^1 \\ \vdots \\ \alpha_i^N \end{pmatrix} = \begin{pmatrix} \beta_1^1 & \cdots & \beta_m^1 \\ \vdots & \ddots & \vdots \\ \beta_1^N & \cdots & \beta_m^N \end{pmatrix} \begin{pmatrix} u_{i,1} \\ \vdots \\ u_{i,m} \end{pmatrix} \tag{8}$$

where $\alpha_i^N$ is $i$-th coefficient of $N$-th frontal image, and $\beta_m^N$ is $m$-th coefficient of $N$-th pose image. This can be written as :

$$A_F = A_L \boldsymbol{u} \tag{9}$$

where

$$A_F = \begin{pmatrix} \alpha_i^1 \\ \vdots \\ \alpha_i^N \end{pmatrix} \tag{10}$$

$$A_L = \begin{pmatrix} \beta_1^1 & \cdots & \beta_m^1 \\ \vdots & \ddots & \vdots \\ \beta_1^N & \cdots & \beta_m^N \\ \end{pmatrix} \tag{11}$$

$$\boldsymbol{u} = \begin{pmatrix} u_{i,1} \\ \vdots \\ u_{i,m} \end{pmatrix} \tag{12}$$

If $A_L^T A_L$ is nonsingular, $\boldsymbol{u}$ is unique and given by

$$\boldsymbol{u} = (A_L^T A_L)^{-1} A_L^T A_F \tag{13}$$

We have m coefficients, accordingly we need to estimate m vectors of $\boldsymbol{u}$ to complete pose transformation matrix $\boldsymbol{U}$. We can obtain pose transformation matrix between input pose and frontal pose using the method we stated. For example, we can write the image of the person at pose L as $I^L = \sum_{i=1}^{m} \alpha_i^L \Phi_i^L$, where $\Phi_i^L$ is $i$-th basis vector for subspace of pose L, and $\alpha_i^L$ is its corresponding coefficient for representing image $I^L$. Since we have pose transformation matrix between left pose and frontal pose $\boldsymbol{U}_{LF}$, we can represent pose transformed version of input image as follow : $I^F = \sum_{i=1}^{m} \boldsymbol{u}_i^T \boldsymbol{\alpha}^L \Phi_i^F$.

## 4 Recognition Methods

After pose transformation, we have images transformed to frontal pose. As a result, we can use algorithms generally used for face recognition such as LDA[5], GDA[6], NDA[7][8], and nearest neighbor for experiments. In this section, we review recognition methods we used for experiments.

### 4.1 LDA and GDA

The face recognition method using LDA is called the fisherface method. It can be applied directly to the gray image[11] [12], or feature vectors of the gray image extracted on a sparse grid[13]. In both cases the classification performance is significantly better than the classification performance of using PCA instead of LDA. LDA is a well-known classical statistical technique using the projection which maximizes the ratio of scatter among the data of different classes to the scatter within the data of the same class[14]. Features obtained by LDA are

useful for pattern classification since they make the data of the same class closer to each other, and the data of different classes further away from each other. Typically, LDA is compared to PCA because both methods are multivariate statistical techniques for projection. PCA attempts to locate the projection that reduces the dimensionality of a data set while retaining as variation in the data set as much as possible[15]. Since PCA does not use class information of data set, LDA usually outperforms PCA for pattern classification.

GDA is nonlinear version of LDA. It generalize LDA to nonlinear problems by mapping the input space into a high dimensional feature space with linear properties. The main idea is to map the input space into a convenient feature space in which variables are nonlinearly related to the input space[6].

### 4.2 NDA

NDA has similar properties with LDA. It seeks the subspace which can effectively discriminate data classes using within and between class scatter matrices. The main difference is that NDA does not make any assumption about the distribution of data, while LDA assumes each class has gaussian distribution. We briefly review the construction of scatter matrices for NDA, detailed explanation of NDA could be found in [7][8].

For computing nonparametric between scatter matrix, we should obtain extra-class nearest neighbor and intra-class nearest neighbor for all sample points. The extra-class nearest neighbor for a sample $x$ of class $C_k$ is defined as $x^E = \{x^{'} \in \overline{C_k} / \|x^{'} - x\| \leq \|z - x\|, \forall z \in \overline{C_k}\}$ and intra-class nearest neighbor is defined as $x^I = \{x^{'} \in C_k / \|x^{'} - x\| \leq \|z - x\|, \forall z \in C_k\}$. From these neighbors, the extra-class difference and intra-class difference is defined as $\triangle^E = x - x^E$ and $\triangle^I = x - x^I$, respectively. Then the nonparametric between scatter matrix is defined as

$$S^E = \frac{1}{N} \sum_{n=1}^{N} w_n (\triangle_n^E)(\triangle_n^E)^T \tag{14}$$

where $N$ is number of all class samples, $\triangle_n^E$ is the extra-class difference for sample $x_n$, and $w_n$ is a sample weight defined as

$$w_n = \frac{min\{\| \triangle_n^E \|^\alpha, \| \triangle_n^I \|^\alpha\}}{\| \triangle_n^E \|^\alpha + \| \triangle_n^I \|^\alpha} \tag{15}$$

where $\alpha$ is a control parameter between zero and infinity. The sample weight controls the influence of samples away from class boundaries.

The within-class scatter matrix is defined the same as LDA.

$$S^I = \frac{1}{K} \sum_{k=1}^{K} \Sigma_k \tag{16}$$


**Fig. 3.** Examples of Database Image

where $\Sigma_k$ is the class-conditional covariance matrix, estimated from the sample set and $K$ is total number of classes.

## 5 Experimental Results

### 5.1 Face Database

We used XM2VT database which consists of 2950 images, i.e. 2 session images of 5 poses for 295 persons. The size of image is $46 \times 56$ and each image is aligned by the eye location. The alignment is performed manually. The database contains 5 pose images for each person as shown in Fig.3. We describe each pose as 'F', 'L', 'R', 'U', 'D' respectively.

### 5.2 Result of Pose Transformation

Fig.4 shows reconstructed pose transformed images of a person. As you can see in Fig.4-(b), transformed images from each pose are similar to each other and they are also similar to the original frontal image. Table.1 represents average reconstruction error of all test images in root-mean squared sense. As expected, the reconstruction error of frontal pose image is the least among 5 poses. But


**Fig. 4.** (a) Original Images (b) Images Transformed to Frontal Pose

**Table 1.** Reconstruction Error

| | FF | RF | LF | UF | DF |
|---|---|---|---|---|---|
| Reconstruction Error | 22.5672 | 30.1874 | 29.9827 | 29.6667 | 30.1382 |

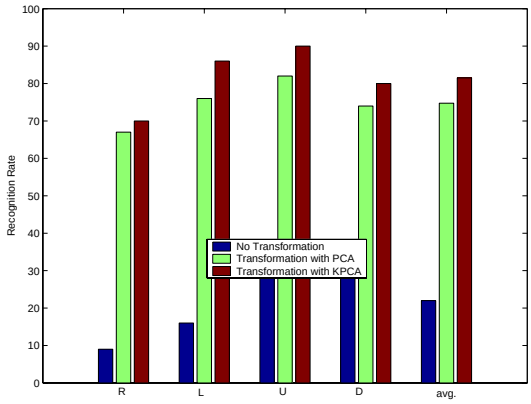


**Fig. 5.** Recognition Rate with Nearest Neighbor Method

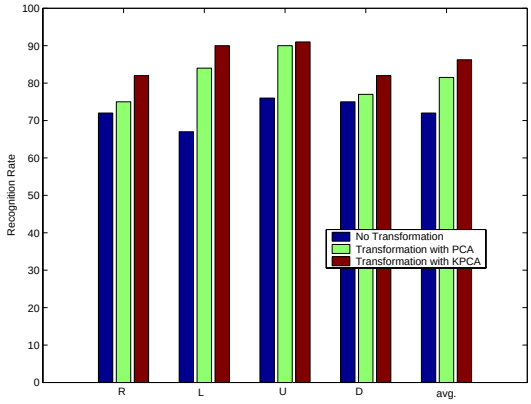


**Fig. 6.** Recognition Rate with LDA

the reconstruction errors of other pose images are also small enough for each pose image to be used for recognition purpose.

### 5.3 Recognition Results with Pose Transformed Image

We used 2450 images of 245 persons for training basis vectors of each pose and pose transformation matrix. In addition, we used 500 images of 50 persons for evaluating the recognition performance of pose transformed images. We used

**Table 2.** Recognition Results

| Pose Transformation? | Recognition Method | RF | LF | UF | DF | Avg. |
|---|---|---|---|---|---|---|
| No Trransformation | NN (in input space) | 9 | 16 | 29 | 34 | 22 |
| | NN (in PCA space with 50 dim.) | 8 | 20 | 23 | 37 | 22 |
| | LDA (in PCA space) | 72 | 67 | 76 | 75 | 72 |
| Pose Transformation with PCA Representation | NN (in PCA space with 50 dim.) | 67 | 76 | 82 | 74 | 74.75 |
| | LDA (in PCA space) | 75 | 84 | 90 | 77 | 81.5 |
| | GDA (in PCA space) | 67 | 82 | 88 | 79 | 79 |
| | NDA (in PCA space) | 72 | 76 | 87 | 79 | 78.5 |
| Pose Transformation with KPCA Representation | NN (in kernel space with 200 dim.) | 70 | 86 | 90 | 80 | 81.5 |
| | LDA (in kernel space) | 82 | 90 | 91 | 82 | 86.25 |
| | GDA (in kernel space) | 77 | 85 | 89 | 83 | 83.5 |
| | NDA (in kernel space) | 85 | 89 | 91 | 91 | 89 |

PCA and Kernel PCA for subspace representation and LDA, GDA, NDA, and nearest neighbor method for face recognition. For recognition experiments, we used two frontal image as gallery and two posed image as probe. For example, RF means that two right posed images are used as probe, two frontal images as gallery.

Table.2 shows the recognition rate. We divided the experiment for 3 parts. First, we checked the recognition rate when no pose transformation is performed. Next, we checked the recognition rate when PCA is used for subspace representation. Finally, we performed the experiment with KPCA as subspace representation. From Fig.5 and Fig.6, you can see recognition rate with pose transformation is much better than no pose transformation case. In addition, we can compare which subspace representation method is better for pose transformation. As you can see in Table.2, recognition rate using kernel PCA is better than PCA case. It is mainly because the nonlinear mapping of KPCA. This makes KPCA represent the input data more effectively than linear PCA. When we use NDA as recognition method, we got the best recognition rate for KPCA case. Because NDA is a nonparametric method, it does not make any assumption about the distribution of data, while LDA assumes each class has gaussian distribution. Moreover, it is uncertain that the transformed frontal pose images of a class are grouped together. Consequently, NDA is a suitable method for this problem. When we used KPCA as subspace representation and NDA as recognition method, we got the best recognition rate.

## 6 Conclusion

In this paper, we proposed linear pose transformation method in feature space. Our method has some merits, since we used 2D appearance based approach instead of using 3D model based method which requires many preprocessing steps, complicated computing steps, and much execution time. We performed various experiments to show the usefulness of proposed pose transformation method for

face recognition. We compared recognition rate with pose transformation and without pose transformation. Moreover, we compared which subspace representation method is better for pose transformation. According to the experimental results, when we use KPCA as subspace representation and NDA as recognition method, we got the best recognition rate.

## References

[1] D.Zhang, D.: AUTOMATED BIOMETRICS-Technologies and Systems. kluwer academic publishers (2000) 211
[2] Beymer, D.: Face recognition under varying pose. In: In Proc. IEEE Conference on Computer Vision and Pattern Recognition. (1994) 756–761 211
[3] Biuk, Z., Loncaric, S.: Face recognition from multi-pose image sequence. In: In Proceedings of 2nd Int'l Symposium on Image and Signal Processing and Analysis. (2001) 211
[4] Huang, F., Zhou, Z., Zhang, H., Chen, T.: Pose invariant face recognition. In: Proceedings of the 4th 1EEE International Conference on Automatic Face and Gesture Recognition. (2000) 245–250 211
[5] Eremad, K., Chellappa, R.: Discriminant analysis for recognition of human face images. Journal of Optical Society of America **14** (1997) 1724–1733 212, 215
[6] Baudat, G., Anouar, F.: Generalized discriminant analysis using a kernel approach. Neural Computation **22** (2000) 2385–2404 212, 215, 216
[7] Bressan, M., Vitria, J.: Nonparametric discriminant analysis and nearest neighbor classification. Pattern Recognition Letters **24** (2003) 2743–2749 212, 215, 216
[8] Fukunaga, K.: Introduction to Statistical Pattern Recognition. Second edn. Academic Press, Boston, MA (1990) 212, 215, 216
[9] Hotelling, H.: Analysis of a compltex statistical variables into principal components. Journal of Educational Psychology **24** (1933) 417–441 212
[10] Scholkopf, B., Smola, A., Muller, K.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10** (1998) 1299–1319 213
[11] Belhumeur, P., Hespanha, J., Kriegman, D.: 1997. eigenfaces vs. fisherfaces: Class specific linear projection. PAMI **19** (1997) 711–720 215
[12] Etemad, K., Chellappa, R.: Discriminant analysis for recognition of human face images. Journal of Optical Society of America A **14** (1997) 1724–1733 215
[13] Duc, B., Fischer, S., Bigun, J.: Face authentication with gabor information on deformable graphs. IEEE Transactions on Image Processing **8** (1999) 504–516 215
[14] Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley, New York (2001) 215
[15] Jolliffe, I.: Principal Component Analysis. Springer-Verlag, New York (1986) 216