POSTECH-MLG-2007-001

Machine Learning Group

Department of Computer Science, POSTECH

Two-Dimensional Canonical Correlation Analysis

Sun Ho Lee and Seungjin Choi

Machine Learning Group
Department of Computer Science
Pohang University of Science and Technology
San 31 Hyoja-dong, Nam-gu
Pohang 790-784, Korea
Email: {sunholee,seungjin}@postech.ac.kr

Abstract

In this technical report we present a method of two-dimensional canonical correlation analysis (2D-CCA) and two-dimensional multiset canonical correlation analysis (2D-MCCA) where we extend the standard CCA and MCCA in such a way that relations between two different sets or more of image data are directly sought without reshaping images into vectors. We stress out that 2D-CCA and 2D-MCCA dramatically reduces the computational complexity, compared to the standard CCA and MCCA. We show the useful behavior of 2D-CCA and 2D-MCCA through numerical examples of correspondence learning between face images in different poses and illumination conditions.

Contents

Machine Learning Group
Department of Computer Science
Pohang University of Science and Technology
San 31 Hyoja-dong, Nam-gu
Pohang 790-784, Korea

1 Introduction

Canonical correlation analysis (CCA) is a multivariate analysis method, the goal of which is to identify and quantify the association between two sets of variables [2]. CCA first seeks a pair of linear combinations (a linear combination of the variables in one set and a linear combination of the variables in another set) which has the largest correlation. Next, it determines a pair of linear combinations having the largest correlation among all pairs uncorrelated with the initially selected pair, and so on. That is, CCA represents a high-dimensional relationship between two sets of variables with a few pairs of canonical variables. Similarly to CCA, MCCA is extension of CCA for multiple sets [3].

During recent years, CCA, MCCA and kernel CCA (KCCA)[4] were successfully applied to content-based retrieval [1], text mining [5, 7], and facial expression recognition [11]. In the case of image data, it is essential to reshape image data into vectors before CCA, MCCA or KCCA is applied. Such reshaping might break the spatial structure of image data and increase the computational complexity.

In this report we present a 2D extension of CCA and MCCA, referred to as 2D-CCA and 2D-MCCA, which directly takes two or more sets of image data as inputs without undergoing reshaping them into vectors, in order to correlate relationships between them. Our 2D-CCA(or 2D-MCCA) was strongly motivated from recent two-dimensional methods such as 2D-PCA [8], GPCA [9], and 2D-LDA [10]. The major difference between CCA and 2D-CCA lies in the data representation. We show that 2D-CCA dramatically reduces the computational complexity, compared to CCA. The rest of this paper is organized as follows. In the next section, we give a brief overview of the standard CCA and MCCA. Sec. 3 and 5 illustrates a 2D extension of CCA and MCCA. Numerical experiments are given in Sec. 6, stressing out the useful behavior of 2D-CCA in correspondence learning between face images in different poses and in illumination conditions. Conclusions are drawn in Sec. 7.

2 CCA

Consider two sets of multivariate data, $\{\boldsymbol{x}_t \in \mathbb{R}^m, t = 1, \ldots, N\}$ and $\{\boldsymbol{y}_t \in \mathbb{R}^n, t = 1, \ldots, N\}$, which are realizations of random vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Mean vectors of $\boldsymbol{x}$ and $\boldsymbol{y}$ are denoted by $\boldsymbol{\mu}_x = \frac{1}{N}\sum_{t=1}^{N} \boldsymbol{x}_t$ and $\boldsymbol{\mu}_y = \frac{1}{N}\sum_{t=1}^{N} \boldsymbol{y}_t$, so that centered data vectors are represented by $\widetilde{\boldsymbol{x}}_t = \boldsymbol{x}_t - \boldsymbol{\mu}_x$ and $\widetilde{\boldsymbol{y}}_t = \boldsymbol{y}_t - \boldsymbol{\mu}_y$, respectively.

CCA seeks a pair of linear transforms, $\boldsymbol{w}_x \in \mathbb{R}^m$ and $\boldsymbol{w}_y \in \mathbb{R}^n$ such that correlations between $\boldsymbol{w}_x^\top \boldsymbol{x}_t$ and $\boldsymbol{w}_y^\top \boldsymbol{y}_t$ are maximized. In other words, the objective function to be maximized is given by

$$\begin{aligned} \rho_1 &= \frac{\operatorname{cov}\left(\boldsymbol{w}_x^\top \boldsymbol{x}, \boldsymbol{w}_y^\top \boldsymbol{y}\right)}{\sqrt{\operatorname{var}\left(\boldsymbol{w}_x^\top \boldsymbol{x}\right) \operatorname{var}\left(\boldsymbol{w}_y^\top \boldsymbol{y}\right)}} \\ &= \frac{\boldsymbol{w}_x^\top \boldsymbol{C}_{xy} \boldsymbol{w}_y}{\sqrt{\left(\boldsymbol{w}_x^\top \boldsymbol{C}_{xx} \boldsymbol{w}_x\right)\left(\boldsymbol{w}_y^\top \boldsymbol{C}_{yy} \boldsymbol{w}_y\right)}}, \end{aligned} \tag{1}$$

where $\boldsymbol{C}_{xy} = \left\langle \widetilde{\boldsymbol{x}} \widetilde{\boldsymbol{y}}^\top \right\rangle = \frac{1}{N}\sum_{t=1}^{N} \widetilde{\boldsymbol{x}}_t \widetilde{\boldsymbol{y}}_t^\top$, where $\langle \cdot \rangle$ denotes the statistical expectation. In a similar manner, $\boldsymbol{C}_{xx} = \left\langle \widetilde{\boldsymbol{x}} \widetilde{\boldsymbol{x}}^\top \right\rangle$ and $\boldsymbol{C}_{yy} = \left\langle \widetilde{\boldsymbol{y}} \widetilde{\boldsymbol{y}}^\top \right\rangle$.

Then, CCA is formulated as

$$\underset{\boldsymbol{w}_x, \boldsymbol{w}_y}{\arg\max}\, \boldsymbol{w}_x^\top \boldsymbol{C}_{xy} \boldsymbol{w}_y, \tag{2}$$

subject to $\boldsymbol{w}_x^\top \boldsymbol{C}_{xx}\boldsymbol{w}_x = 1$ and $\boldsymbol{w}_y^\top \boldsymbol{C}_{yy}\boldsymbol{w}_y = 1$. Reflecting these two constraints, the Lagrangian $\mathcal{J}$ is given by

$$\begin{aligned}\mathcal{J} &= \boldsymbol{w}_x^\top \boldsymbol{C}_{xy}\boldsymbol{w}_y + \lambda_x\left(1 - \boldsymbol{w}_x^\top \boldsymbol{C}_{xx}\boldsymbol{w}_x\right) \\ &\quad +\lambda_y\left(1 - \boldsymbol{w}_y^\top \boldsymbol{C}_{yy}\boldsymbol{w}_y\right). \end{aligned} \tag{3}$$

It is well known that $\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}_x} = 0$ and $\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}_y} = 0$ leads to the following generalized eigenvalue problem

$$\begin{bmatrix} 0 & \boldsymbol{C}_{xy} \\ \boldsymbol{C}_{yx} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_x \\ \boldsymbol{w}_y \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{C}_{xx} & 0 \\ 0 & \boldsymbol{C}_{yy} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_x \\ \boldsymbol{w}_y \end{bmatrix}, \tag{4}$$

where $\lambda = 2\lambda_x = 2\lambda_y$. Those readers who are not familiar with CCA, can easily see how this generalized eigenvalue problem emerges from the formulation (2), from the appendix where we show detailed derivations for 2D-CCA.

3 2D-CCA

Now we consider two sets of image data, $\{\boldsymbol{X}_t \in \mathbb{R}^{m_x \times n_x},\ t = 1, \ldots, N\}$ and $\{\boldsymbol{Y}_t \in \mathbb{R}^{m_y \times n_y},\ t = 1, \ldots, N\}$ which are realizations of random variable matrix $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. In the conventional CCA, each image data is reshaped into a long vector, then the method described in Sec. 2 is applied. Here we present 2D-CCA where we directly use image data to determine relations between them.

We define mean matrices of $\boldsymbol{X}_t$ and $\boldsymbol{Y}_t$ as

$$\begin{aligned}\boldsymbol{M}_x &= \frac{1}{N}\sum_{t=1}^{N} \boldsymbol{X}_t, \\ \boldsymbol{M}_y &= \frac{1}{N}\sum_{t=1}^{N} \boldsymbol{Y}_t.\end{aligned}$$

Then centered image data are denoted by

$$\begin{aligned}\widetilde{\boldsymbol{X}}_t &= \boldsymbol{X}_t - \boldsymbol{M}_x, \\ \widetilde{\boldsymbol{Y}}_t &= \boldsymbol{Y}_t - \boldsymbol{M}_y.\end{aligned}$$

2D-CCA seeks left transforms $\boldsymbol{l}_x$ and $\boldsymbol{l}_y$ and right transforms $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$ such that correlations between $\boldsymbol{l}_x^\top \boldsymbol{X}\boldsymbol{r}_x$ and $\boldsymbol{l}_y^\top \boldsymbol{Y}\boldsymbol{r}_y$ are maximized. Then 2D CCA is formulated as

$$\begin{aligned}\underset{\boldsymbol{l}_x, \boldsymbol{r}_x, \boldsymbol{l}_y, \boldsymbol{r}_y}{\arg\max}\ & \operatorname{cov}\left(\boldsymbol{l}_x^\top \boldsymbol{X}\boldsymbol{r}_x, \boldsymbol{l}_y^\top \boldsymbol{Y}\boldsymbol{r}_y\right), \\ \text{s.t.}\ & \operatorname{var}\left(\boldsymbol{l}_x^\top \boldsymbol{X}\boldsymbol{r}_x\right) = 1, \\ & \operatorname{var}\left(\boldsymbol{l}_y^\top \boldsymbol{Y}\boldsymbol{r}_y\right) = 1. \end{aligned} \tag{5}$$

We define

$$\boldsymbol{\Sigma}_{xy}^r = \left\langle \widetilde{\boldsymbol{X}}\boldsymbol{r}_x\boldsymbol{r}_y^\top \widetilde{\boldsymbol{Y}}^\top \right\rangle = \frac{1}{N}\sum_{t=1}^{N} \widetilde{\boldsymbol{X}}_t\boldsymbol{r}_x\boldsymbol{r}_y^\top \widetilde{\boldsymbol{Y}}_t^\top, \tag{6}$$

$$\boldsymbol{\Sigma}_{xx}^r = \left\langle \widetilde{\boldsymbol{X}}\boldsymbol{r}_x\boldsymbol{r}_x^\top \widetilde{\boldsymbol{X}}^\top \right\rangle = \frac{1}{N}\sum_{t=1}^{N} \widetilde{\boldsymbol{X}}_t\boldsymbol{r}_x\boldsymbol{r}_x^\top \widetilde{\boldsymbol{X}}_t^\top, \tag{7}$$

$$\boldsymbol{\Sigma}_{yy}^r = \left\langle \widetilde{\boldsymbol{Y}}\boldsymbol{r}_y\boldsymbol{r}_y^\top \widetilde{\boldsymbol{Y}}^\top \right\rangle = \frac{1}{N}\sum_{t=1}^{N} \widetilde{\boldsymbol{Y}}_t\boldsymbol{r}_r\boldsymbol{r}_y^\top \widetilde{\boldsymbol{Y}}_t^\top. \tag{8}$$

Note that we can write

$$\begin{aligned}\operatorname{cov}\left(\boldsymbol{l}_x^\top \boldsymbol{X}\boldsymbol{r}_x, \boldsymbol{l}_y^\top \boldsymbol{Y}\boldsymbol{r}_y\right) &= \left\langle \boldsymbol{l}_x^\top \widetilde{\boldsymbol{X}}\boldsymbol{r}_x \boldsymbol{r}_y^\top \widetilde{\boldsymbol{Y}}^\top \boldsymbol{l}_y \right\rangle \\ &= \boldsymbol{l}_x^\top \boldsymbol{\Sigma}_{xy}^r \boldsymbol{l}_y.\end{aligned}$$

With these definitions, the formulation of 2D CCA given in (5) is rewritten as

$$\begin{aligned}&\arg\max \boldsymbol{l}_x^\top \boldsymbol{\Sigma}_{xy}^r \boldsymbol{l}_y, \\ &\text{s.t. } \boldsymbol{l}_x^\top \boldsymbol{\Sigma}_{xx}^r \boldsymbol{l}_x = 1, \\ &\quad \boldsymbol{l}_y^\top \boldsymbol{\Sigma}_{yy}^r \boldsymbol{l}_y = 1.\end{aligned} \tag{9}$$

Note also that

$$\operatorname{cov}\left(\boldsymbol{l}_x^\top \boldsymbol{X}\boldsymbol{r}_x, \boldsymbol{l}_y^\top \boldsymbol{Y}\boldsymbol{r}_y\right) = \operatorname{cov}\left(\boldsymbol{r}_x^\top \boldsymbol{X}^\top \boldsymbol{l}_x, \boldsymbol{r}_y^\top \boldsymbol{Y}^\top \boldsymbol{l}_y\right).$$

Hence, alternatively, we can also rewrite (5) as

$$\begin{aligned}&\arg\max \boldsymbol{r}_x^\top \boldsymbol{\Sigma}_{xy}^l \boldsymbol{r}_y, \\ &\text{s.t. } \boldsymbol{r}_x^\top \boldsymbol{\Sigma}_{xx}^l \boldsymbol{r}_x = 1 \\ &\quad \boldsymbol{r}_y^\top \boldsymbol{\Sigma}_{yy}^l \boldsymbol{r}_y = 1,\end{aligned} \tag{10}$$

where

$$\boldsymbol{\Sigma}_{xy}^l = \left\langle \widetilde{\boldsymbol{X}}^\top \boldsymbol{l}_x \boldsymbol{l}_y^\top \widetilde{\boldsymbol{Y}} \right\rangle, \tag{11}$$

$$\boldsymbol{\Sigma}_{xx}^l = \left\langle \widetilde{\boldsymbol{X}}^\top \boldsymbol{l}_x \boldsymbol{l}_x^\top \widetilde{\boldsymbol{X}} \right\rangle, \tag{12}$$

$$\boldsymbol{\Sigma}_{yy}^l = \left\langle \widetilde{\boldsymbol{Y}}^\top \boldsymbol{l}_y \boldsymbol{l}_y^\top \widetilde{\boldsymbol{Y}} \right\rangle. \tag{13}$$

2D-CCA determines $\boldsymbol{l}_x$ and $\boldsymbol{l}_y$ by solving (9) with $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$ fixed. The right transforms $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$ are found by solving (10) with $\boldsymbol{l}_x$ and $\boldsymbol{l}_y$ fixed. Given $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$, the optimization in (9) involves the following generalized eigenvalue problem (see Appendix for detailed derivation):

$$\begin{bmatrix} 0 & \boldsymbol{\Sigma}_{xy}^r \\ \boldsymbol{\Sigma}_{yx}^r & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{l}_x \\ \boldsymbol{l}_y \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{\Sigma}_{xx}^r & 0 \\ 0 & \boldsymbol{\Sigma}_{yy}^r \end{bmatrix} \begin{bmatrix} \boldsymbol{l}_x \\ \boldsymbol{l}_y \end{bmatrix}. \tag{14}$$

In a similar manner, given $\boldsymbol{l}_x$ and $\boldsymbol{l}_y$, the optimization in (10) involves the following generalized eigenvalue problem:

$$\begin{bmatrix} 0 & \boldsymbol{\Sigma}_{xy}^l \\ \boldsymbol{\Sigma}_{yx}^l & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{r}_y \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{\Sigma}_{xx}^l & 0 \\ 0 & \boldsymbol{\Sigma}_{yy}^l \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{r}_y \end{bmatrix}. \tag{15}$$

Left transforms ($\boldsymbol{l}_x$ and $\boldsymbol{l}_y$) and right transforms ($\boldsymbol{r}_x$ and $\boldsymbol{r}_y$) are determined by iteratively solving (14) and (15) until convergence (see Table 1 for the outline of 2D-CCA algorithm). In our numerical experiments, it takes only a few iterations for convergence. The d_1 largest generalized eigenvectors in (14) determines $\boldsymbol{L}_x \in \mathbb{R}^{m_x \times d_1}$ and $\boldsymbol{L}_y \in \mathbb{R}^{m_y \times d_2}$. In a similar manner, the d_2 largest generalized eigenvectors in (15) gives $\boldsymbol{R}_x \in \mathbb{R}^{n_x \times d_2}$, $\boldsymbol{R}_y \in \mathbb{R}^{n_y \times d_2}$. In the case of image data, 2D-CCA involves the generalized eigenvalue problem for much smaller size matrices, compared to the standard CCA, which reduces the complexity dramatically.

Table 1: Algorithm outline: 2D-CCA

Input
- Image data $\{\boldsymbol{X}_t \in \mathbb{R}^{m_x \times n_x}\}_{t=1}^N$ and $\{\boldsymbol{Y}_t \in \mathbb{R}^{m_y \times n_y}\}_{t=1}^N$
- The size of the canonical variable matrix: $d_1 \times d_2$
Output
- Left transforms $\boldsymbol{L}_x \in \mathbb{R}^{m_x \times d_1}, \boldsymbol{L}_y \in \mathbb{R}^{m_y \times d_1}$
- Right transforms $\boldsymbol{R}_x \in \mathbb{R}^{n_x \times d_2}, \boldsymbol{R}_y \in \mathbb{R}^{n_y \times d_2}$
Do centering image data to get $\{\widetilde{\boldsymbol{X}}_t\}$ and $\{\widetilde{\boldsymbol{Y}}_t\}$
Initialize $\boldsymbol{R}_x$ and $\boldsymbol{R}_y$
repeat
Compute the following matrices
$\boldsymbol{\Sigma}_{xx}^r = \frac{1}{N}\sum_{t=1}^N \widetilde{\boldsymbol{X}}_t \boldsymbol{R}_x \boldsymbol{R}_x^\top \widetilde{\boldsymbol{X}}_t^\top$,
$\boldsymbol{\Sigma}_{xy}^r = \frac{1}{N}\sum_{t=1}^N \widetilde{\boldsymbol{X}}_t \boldsymbol{R}_x \boldsymbol{R}_y^\top \widetilde{\boldsymbol{Y}}_t^\top$, $\boldsymbol{\Sigma}_{yx}^r = \left[\boldsymbol{\Sigma}_{xy}^r\right]^\top$,
$\boldsymbol{\Sigma}_{yy}^r = \frac{1}{N}\sum_{t=1}^N \widetilde{\boldsymbol{Y}}_t \boldsymbol{R}_y \boldsymbol{R}_y^\top \widetilde{\boldsymbol{Y}}_t^\top$.
Compute d_1 largest generalized eigenvectors in (14): $\boldsymbol{L}_x$ and $\boldsymbol{L}_y$
Compute the following matrices
$\boldsymbol{\Sigma}_{xx}^l = \frac{1}{N}\sum_{t=1}^N \widetilde{\boldsymbol{X}}_t^\top \boldsymbol{L}_x \boldsymbol{L}_x^\top \widetilde{\boldsymbol{X}}_t$,
$\boldsymbol{\Sigma}_{xy}^l = \frac{1}{N}\sum_{t=1}^N \widetilde{\boldsymbol{X}}_t^\top \boldsymbol{L}_x \boldsymbol{L}_y^\top \widetilde{\boldsymbol{Y}}_t$, $\boldsymbol{\Sigma}_{yx}^l = \left[\boldsymbol{\Sigma}_{xy}^l\right]^\top$,
$\boldsymbol{\Sigma}_{yy}^l = \frac{1}{N}\sum_{t=1}^N \widetilde{\boldsymbol{Y}}_t^\top \boldsymbol{L}_y \boldsymbol{L}_y^\top \widetilde{\boldsymbol{Y}}_t$.
Compute d_2 largest generalized eigenvectors in (15): $\boldsymbol{R}_x$ and $\boldsymbol{R}_y$
until (converged)

4 MCCA

Consider an $m = m_1 + m_2 + \cdots + m_n$ dimensional random variable $\boldsymbol{x}$ which follows a Gaussian distribution divide into n groups of dimensions, $\{\ \boldsymbol{x}_i \in \mathbb{R}^{m_i}, i = 1, \cdots, n\}$.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = N\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_n \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} & \cdots & \boldsymbol{\Sigma}_{1n} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} & \cdots & \boldsymbol{\Sigma}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1} & \boldsymbol{\Sigma}_{n2} & \cdots & \boldsymbol{\Sigma}_{nn} \end{bmatrix}\right)$$

In this case, we assume that the covariance matrices are non-singular and $\boldsymbol{\mu}_i = \boldsymbol{0}$. Our aim is to seek a set of linear combinations, $\boldsymbol{w}_i \in \mathbb{R}^{m_i}$

$$\begin{aligned} \boldsymbol{u}_1 &= \boldsymbol{w}_1^\top \boldsymbol{x}_1 \\ \boldsymbol{u}_2 &= \boldsymbol{w}_2^\top \boldsymbol{x}_2 \\ &\vdots \\ \boldsymbol{u}_n &= \boldsymbol{w}_n^\top \boldsymbol{x}_n \end{aligned}$$

such that the variance $\mathrm{V} = V\{\boldsymbol{w}^\top \boldsymbol{x}\} = \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}$ is maximized, where $\boldsymbol{w}^\top = [\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \cdots, \boldsymbol{w}_n^\top]$. In other words, the sum of elements in covariance matrix of transformed variables are maximized, i.e, maximization of $\sum_{i=1}^n \sum_{j=1}^n \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ij} \boldsymbol{w}_j$ (Kettenring named this as SUMCOR). In this situation, several natrual constraints can be applied to perform optimizations.

1. $\boldsymbol{w}_i^\top \boldsymbol{w}_i = 1$

2. $\sum_{i=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{w}_i = 1$
3. $\boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ii} \boldsymbol{w}_i = 1$
4. $\sum_{i=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ii} \boldsymbol{w}_i = 1$

In case of two data sets, SUMCOR with constraint 3 and 4 can be reduced to the standard Hotelling method. In order to reflect these constraints, we use a Lagrange multiplier.

Constraint 1 : $\boldsymbol{w}_i^\top \boldsymbol{w}_i = 1$
The Lagrangian $\mathcal{J}$ is given by

$$\mathcal{J} = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ij} \boldsymbol{w}_j + \sum_{i=1}^{n} \lambda_i (1 - \boldsymbol{w}_i^\top \boldsymbol{w}_i) \tag{16}$$

Solving $\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}_i} = 0$ leads to

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} & \cdots & \boldsymbol{\Sigma}_{1n} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} & \cdots & \boldsymbol{\Sigma}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1} & \boldsymbol{\Sigma}_{n2} & \cdots & \boldsymbol{\Sigma}_{nn} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_n \end{bmatrix} = \begin{bmatrix} \lambda_1 \boldsymbol{w}_1 \\ \lambda_2 \boldsymbol{w}_2 \\ \vdots \\ \lambda_n \boldsymbol{w}_n \end{bmatrix} \tag{17}$$

This is not eigen-value problem.

Constraint 2 : $\sum_{i=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{w}_i = 1$
Introduce

$$\mathcal{J} = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ij} \boldsymbol{w}_j + \lambda (1 - \sum_{i=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{w}_i) \tag{18}$$

By setting $\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}_i} = 0$, we acquire

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} & \cdots & \boldsymbol{\Sigma}_{1n} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} & \cdots & \boldsymbol{\Sigma}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1} & \boldsymbol{\Sigma}_{n2} & \cdots & \boldsymbol{\Sigma}_{nn} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_n \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_n \end{bmatrix} \tag{19}$$

This is normal real, symmetric eigen-value problem.

Constraint 3 : $\boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ii} \boldsymbol{w}_i = 1$

$$\mathcal{J} = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ij} \boldsymbol{w}_j + \sum_{i=1}^{n} \lambda_i (1 - \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ii} \boldsymbol{w}_i) \tag{20}$$

Solving $\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}_i} = 0$ leads to

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} & \cdots & \boldsymbol{\Sigma}_{1n} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} & \cdots & \boldsymbol{\Sigma}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1} & \boldsymbol{\Sigma}_{n2} & \cdots & \boldsymbol{\Sigma}_{nn} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_n \end{bmatrix} = \begin{bmatrix} \lambda_1 \boldsymbol{\Sigma}_{11} & 0 & \cdots & 0 \\ 0 & \lambda_2 \boldsymbol{\Sigma}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \boldsymbol{\Sigma}_{nn} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_n \end{bmatrix} \tag{21}$$

This is not a normal generalized eigen-value problem.

Constraint 4 : $\sum_{i=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ii} \boldsymbol{w}_i = 1$

$$\mathcal{J} = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ij} \boldsymbol{w}_j + \lambda(1 - \sum_{i=1}^{n} \boldsymbol{w}_i^\top \boldsymbol{\Sigma}_{ii} \boldsymbol{w}_i) \tag{22}$$

Setting $\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}_i} = 0$ leads to

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} & \cdots & \boldsymbol{\Sigma}_{1n} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} & \cdots & \boldsymbol{\Sigma}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1} & \boldsymbol{\Sigma}_{n2} & \cdots & \boldsymbol{\Sigma}_{nn} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_n \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{\Sigma}_{11} & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Sigma}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\Sigma}_{nn} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_n \end{bmatrix} \tag{23}$$

This is a generalized eigen-value problem.

5 Two-Dimensional MCCA

Now we consider n sets of image data, $\{\ \boldsymbol{X}_i \in \mathbb{R}^{r_i \times c_i}, i = 1, \cdots, n\ \}$. In the conventional MCCA, each image data is reshaped into a long vector, then use the method described in Sec. 4. Here we address 2D-MCCA where we directly use image data to determine relations between them. Analogously to MCCA, 2D-MCCA seeks left transforms $\boldsymbol{l}_i$ and right transforms $\boldsymbol{r}_i$ $\{\boldsymbol{l}_i \in \mathbb{R}^{r_i}, \boldsymbol{r}_i \in \mathbb{R}^{c_i}, i = 1, \cdots, n\}$

$$\boldsymbol{U}_1 = \boldsymbol{l}_1^\top \boldsymbol{X}_1 \boldsymbol{r}_1,\ \mathrm{V}\{\boldsymbol{U}_1\} = \boldsymbol{l}_1^\top \boldsymbol{\Sigma}_{11}^r \boldsymbol{l}_1 \quad \text{or} \quad \mathrm{V}\{\boldsymbol{U}_1\} = \boldsymbol{r}_1^\top \boldsymbol{\Sigma}_{11}^l \boldsymbol{r}_1$$

$$\boldsymbol{U}_2 = \boldsymbol{l}_2^\top \boldsymbol{X}_2 \boldsymbol{r}_2,\ \mathrm{V}\{\boldsymbol{U}_2\} = \boldsymbol{l}_2^\top \boldsymbol{\Sigma}_{22}^r \boldsymbol{l}_2 \quad \text{or} \quad \mathrm{V}\{\boldsymbol{U}_2\} = \boldsymbol{r}_2^\top \boldsymbol{\Sigma}_{22}^l \boldsymbol{r}_2$$

$$\vdots$$

$$\boldsymbol{U}_n = \boldsymbol{l}_n^\top \boldsymbol{X}_n \boldsymbol{r}_n,\ \mathrm{V}\{\boldsymbol{U}_n\} = \boldsymbol{l}_n^\top \boldsymbol{\Sigma}_{nn}^r \boldsymbol{l}_n \quad \text{or} \quad \mathrm{V}\{\boldsymbol{U}_n\} = \boldsymbol{r}_n^\top \boldsymbol{\Sigma}_{nn}^l \boldsymbol{r}_n$$

with maximum variance

$$\begin{aligned} V &= \mathrm{V}\{\sum_{i=1}^{n} \boldsymbol{l}_i^\top \boldsymbol{X}_i \boldsymbol{r}_i\} \\ &= \mathrm{V}\{\boldsymbol{l}_1^\top \boldsymbol{X}_1 \boldsymbol{r}_1 + \boldsymbol{l}_2^\top \boldsymbol{X}_2 \boldsymbol{r}_2 + \cdots \boldsymbol{l}_n^\top \boldsymbol{X}_n \boldsymbol{r}_n\} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{l}_i^\top \boldsymbol{\Sigma}_{ij}^r \boldsymbol{l}_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{r}_i^\top \boldsymbol{\Sigma}_{ij}^l \boldsymbol{r}_j. \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{\Sigma}_{nn}^r &= \left\langle \widetilde{\boldsymbol{X}_n} \boldsymbol{r}_n \boldsymbol{r}_n^\top \widetilde{\boldsymbol{X}_n}^\top \right\rangle, \\ \boldsymbol{\Sigma}_{nn}^l &= \left\langle \widetilde{\boldsymbol{X}_n}^\top \boldsymbol{l}_n \boldsymbol{l}_n^\top \widetilde{\boldsymbol{X}_n} \right\rangle. \end{aligned}$$

In other words, we can see that maximizing the above variance corresponds to maximizing the sum of the elements(SUMCOR) in the covariance matrix of the transformed variables

$$\boldsymbol{\Sigma}_U^r = \begin{bmatrix} \boldsymbol{l}_1^\top \boldsymbol{\Sigma}_{11}^r \boldsymbol{l}_1 & \boldsymbol{l}_1^\top \boldsymbol{\Sigma}_{12}^r \boldsymbol{l}_2 & \cdots & \boldsymbol{l}_1^\top \boldsymbol{\Sigma}_{1n}^r \boldsymbol{l}_n \\ \boldsymbol{l}_2^\top \boldsymbol{\Sigma}_{21}^r \boldsymbol{l}_1 & \boldsymbol{l}_2^\top \boldsymbol{\Sigma}_{22}^r \boldsymbol{l}_2 & \cdots & \boldsymbol{l}_2^\top \boldsymbol{\Sigma}_{2n}^r \boldsymbol{l}_n \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{l}_n^\top \boldsymbol{\Sigma}_{n1}^r \boldsymbol{l}_1 & \boldsymbol{l}_n^\top \boldsymbol{\Sigma}_{n2}^r \boldsymbol{l}_2 & \cdots & \boldsymbol{l}_n^\top \boldsymbol{\Sigma}_{nn}^r \boldsymbol{l}_n \end{bmatrix} \tag{24}$$

or

$$\boldsymbol{\Sigma}_U^l = \begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{\Sigma}_{11}^l \boldsymbol{r}_1 & \boldsymbol{r}_1^\top \boldsymbol{\Sigma}_{12}^l \boldsymbol{r}_2 & \cdots & \boldsymbol{r}_1^\top \boldsymbol{\Sigma}_{1n}^l \boldsymbol{r}_n \\ \boldsymbol{r}_2^\top \boldsymbol{\Sigma}_{21}^l \boldsymbol{r}_1 & \boldsymbol{r}_2^\top \boldsymbol{\Sigma}_{22}^l \boldsymbol{r}_2 & \cdots & \boldsymbol{r}_2^\top \boldsymbol{\Sigma}_{2n}^l \boldsymbol{r}_n \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{r}_n^\top \boldsymbol{\Sigma}_{n1}^l \boldsymbol{r}_1 & \boldsymbol{r}_n^\top \boldsymbol{\Sigma}_{n2}^l \boldsymbol{r}_2 & \cdots & \boldsymbol{r}_n^\top \boldsymbol{\Sigma}_{nn}^l \boldsymbol{r}_n \end{bmatrix} \tag{25}$$

where,

$$\begin{aligned} \boldsymbol{\Sigma}_{ij}^r &= \left\langle \widetilde{\boldsymbol{X}}_i \boldsymbol{r}_i \boldsymbol{r}_j^\top \widetilde{\boldsymbol{X}_j}^\top \right\rangle, \\ \boldsymbol{\Sigma}_{ij}^l &= \left\langle \widetilde{\boldsymbol{X}_i}^\top \boldsymbol{l}_i \boldsymbol{l}_j^\top \widetilde{\boldsymbol{X}_j} \right\rangle. \end{aligned}$$

Similarly to MCCA, two constraints can be applied to (24) or (25).

Constraint :

$$1. \quad \sum_{i=1}^n \boldsymbol{l}_i^\top \boldsymbol{\Sigma}_{ii}^r \boldsymbol{l}_i = \mathrm{tr}\boldsymbol{\Sigma}_U^r = 1 \quad \text{or} \quad \sum_{i=1}^n \boldsymbol{r}_i^\top \boldsymbol{\Sigma}_{ii}^l \boldsymbol{r}_i = \mathrm{tr}\boldsymbol{\Sigma}_U^l = 1 \tag{26}$$

$$2. \quad \sum_{i=1}^n \boldsymbol{l}_i^\top \boldsymbol{l}_i = 1 \quad \text{or} \quad \sum_{i=1}^n \boldsymbol{r}_i^\top \boldsymbol{r}_i = 1 \tag{27}$$

2D-MCCA determines $\boldsymbol{l}_i$ $(i = 1, \cdots, n)$ by solving SUMCOR with constraint (26), (27) and fixed $\boldsymbol{r}_i$ $(i = 1, \cdots, n)$, or $\boldsymbol{r}_i$ by solving SUMCOR with constraint (26), (27) and fixed $\boldsymbol{l}_i$ $(i = 1, \cdots, n)$

5.1 Solution for constraint 1

In this section, we derive algorithm using (26). First, in order to get solution for fixed $\boldsymbol{r}_i$ $(i = 1, \cdots, n)$, we can formulate objective function as follow.

$$\mathcal{J}_l = \sum_{i=1}^n \sum_{j=1}^n \boldsymbol{l}_i^\top \boldsymbol{\Sigma}_{ij}^r \boldsymbol{l}_j + \lambda \left(1 - \sum_{i=1}^n \boldsymbol{l}_i^\top \boldsymbol{\Sigma}_{ii}^r \boldsymbol{l}_i \right) \tag{28}$$

Solving $\frac{\partial \mathcal{J}_l}{\partial \boldsymbol{l}_i} = 0$ leads to

$$\sum_{j=1}^n \boldsymbol{\Sigma}_{ij}^r \boldsymbol{l}_j - \lambda \boldsymbol{\Sigma}_{ii}^r \boldsymbol{l}_i = 0 \tag{29}$$

From (29), we can rewrite equations as follows

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11}^r & \boldsymbol{\Sigma}_{12}^r & \cdots & \boldsymbol{\Sigma}_{1n}^r \\ \boldsymbol{\Sigma}_{21}^r & \boldsymbol{\Sigma}_{22}^r & \cdots & \boldsymbol{\Sigma}_{2n}^r \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1}^r & \boldsymbol{\Sigma}_{n2}^r & \cdots & \boldsymbol{\Sigma}_{nn}^r \end{bmatrix} \begin{bmatrix} \boldsymbol{l}_1 \\ \boldsymbol{l}_2 \\ \vdots \\ \boldsymbol{l}_n \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{\Sigma}_{11}^r & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Sigma}_{22}^r & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\Sigma}_{nn}^r \end{bmatrix} \begin{bmatrix} \boldsymbol{l}_1 \\ \boldsymbol{l}_2 \\ \vdots \\ \boldsymbol{l}_n \end{bmatrix} \tag{30}$$

This is a generalized eigenvalue problem.
In a similar manner, given $\boldsymbol{l}_i$, we can formulate objective function.

$$\mathcal{J}_r = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{r}_i^\top \boldsymbol{\Sigma}_{ij}^l \boldsymbol{r}_j + \lambda\left(1 - \sum_{i=1}^{n} \boldsymbol{r}_i^\top \boldsymbol{\Sigma}_{ii}^l \boldsymbol{r}_i\right) \tag{31}$$

Solving $\frac{\partial \mathcal{J}_r}{\partial \boldsymbol{r}_i} = 0$ leads to

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11}^l & \boldsymbol{\Sigma}_{12}^l & \cdots & \boldsymbol{\Sigma}_{1n}^l \\ \boldsymbol{\Sigma}_{21}^l & \boldsymbol{\Sigma}_{22}^l & \cdots & \boldsymbol{\Sigma}_{2n}^l \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1}^l & \boldsymbol{\Sigma}_{n2}^l & \cdots & \boldsymbol{\Sigma}_{nn}^l \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_n \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{\Sigma}_{11}^l & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Sigma}_{22}^l & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\Sigma}_{nn}^l \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_n \end{bmatrix} \tag{32}$$

From these equations, Left transforms $(\boldsymbol{l}_i, i = 1, \cdots, n)$ and right transforms $(\boldsymbol{r}_i, i = 1, \cdots, n)$ are determined by iteratively solving (30) and (32) until convergence. The d_1 largest generalized eigenvectors in (30) determine $\boldsymbol{L}_i \in \mathbb{R}^{r_i \times d_1}$. In a similar manner, the d_2 largest generalized eigenvectors in (32) give $\boldsymbol{R}_i \in \mathbb{R}^{c_i \times d_2}$. In the case of image data , 2D-MCCA involves the generalized eigenvalue problem for much smaller size matrices, compared to the conventional MCCA, which reduces the complexity dramatically.

5.2 Solution for constraint 2

In this section, we solve SUMCOR with (27). In a similar way of subsection 5.1, for fixed $\boldsymbol{r}_i$ $(i = 1, \cdots, n)$, we can formulate objective function.

$$\mathcal{J}_l = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{l}_i^\top \boldsymbol{\Sigma}_{ij}^r \boldsymbol{l}_j + \lambda\left(1 - \sum_{i=1}^{n} \boldsymbol{l}_i^\top \boldsymbol{l}_i\right) \tag{33}$$

Solving $\frac{\partial \mathcal{J}_l}{\partial \boldsymbol{l}_i} = 0$ leads to

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11}^r & \boldsymbol{\Sigma}_{12}^r & \cdots & \boldsymbol{\Sigma}_{1n}^r \\ \boldsymbol{\Sigma}_{21}^r & \boldsymbol{\Sigma}_{22}^r & \cdots & \boldsymbol{\Sigma}_{2n}^r \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1}^r & \boldsymbol{\Sigma}_{n2}^r & \cdots & \boldsymbol{\Sigma}_{nn}^r \end{bmatrix} \begin{bmatrix} \boldsymbol{l}_1 \\ \boldsymbol{l}_2 \\ \vdots \\ \boldsymbol{l}_n \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{l}_1 \\ \boldsymbol{l}_2 \\ \vdots \\ \boldsymbol{l}_n \end{bmatrix} \tag{34}$$

Similarly, given $\boldsymbol{l}_i$, the objective function can be formulated.

$$\mathcal{J}_r = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{r}_i^\top \boldsymbol{\Sigma}_{ij}^l \boldsymbol{r}_j + \lambda\left(1 - \sum_{i=1}^{n} \boldsymbol{r}_i^\top \boldsymbol{r}_i\right) \tag{35}$$

Solving $\frac{\partial \mathcal{J}_r}{\partial \boldsymbol{r}_i} = 0$ leads to

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11}^l & \boldsymbol{\Sigma}_{12}^l & \cdots & \boldsymbol{\Sigma}_{1n}^l \\ \boldsymbol{\Sigma}_{21}^l & \boldsymbol{\Sigma}_{22}^l & \cdots & \boldsymbol{\Sigma}_{2n}^l \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{n1}^l & \boldsymbol{\Sigma}_{n2}^l & \cdots & \boldsymbol{\Sigma}_{nn}^l \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_n \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_n \end{bmatrix} \tag{36}$$

From these equations, Left transforms $(\boldsymbol{l}_i, i = 1, \cdots, n)$ and right transforms $(\boldsymbol{r}_i, i = 1, \cdots, n)$ are determined by iteratively solving (34) and (36) until convergence. The d_1 largest generalized eigenvectors in (34) determine $\boldsymbol{L}_i \in \mathbb{R}^{r_i \times d_1}$. In a similar manner, the d_2 largest generalized eigenvectors in (36) give $\boldsymbol{R}_i \in \mathbb{R}^{c_i \times d_2}$.

6 Numerical Experiments

We present two numerical experiments with AR and FERET face DB. Experiment 1 (with AR face DB) involves the application of CCA methods (including 2D methods) for correspondence learning between face images in different illumination conditions. In Experiment 2, we use FERET face DB, investigating the behavior of correspondence learning between face images in different poses. All experiments were carried out on a PC with P4 3.4G CPU with 2G RAM. In case of 2D-MCCA, we assumed that we already know the groups of test data in all experiments.

6.1 Experiment 1: Face images in different illuminations

We collected 536 face images of 134 people (75 male and 59 female) from AR face DB. For each person, there are 4 different face images, three of which involves different illumination conditions. Fig. 1 shows exemplary face images used in our experiment. Each face image was resized by 50×50 and was normalized according to the eye location.

Figure 1: Exemplary face images in AR DB are shown. Face images in the top row are references without varying illumination conditions and the rest of images are faces taken in different illumination conditions.

We partition the dataset into two groups where one group contains reference face images and the other group contains images in different illumination conditions. The first group $\{\boldsymbol{X}_t\}$ consists of 402 reference face images (3 copies of 134 face images in order to balance the number of samples with the second group containing images in 3 different illumination conditions). The second group $\{\boldsymbol{Y}_t\}$ consists of 402 face images in different illumination conditions.

The task here is to investigate how well CCA or 2D-CCA can relate face images in different illumination conditions to corresponding reference face images. In such a recognition task, we use 1-NN classifier. For CCA, images are reshaped into 2500-dimensional vectors and d-dimensional canonical correlation variable vectors ($\boldsymbol{W}_x^\top \boldsymbol{x}$ and $\boldsymbol{W}_y^\top \boldsymbol{y}$) are used in classification. In the case of 2D-CCA, images are directly used and $d \times d$ canonical correlation variable matrices ($\boldsymbol{L}_x^\top \boldsymbol{X} \boldsymbol{R}_x$ and $\boldsymbol{L}_y^\top \boldsymbol{Y} \boldsymbol{R}_y$) are used as inputs to 1-NN classifier. With 10-fold cross-validation, the averaged accuracy is summarized in Table 2, where the results of regularized LDA (rLDA) [6] and 2D-LDA [10] are also shown. In cases of rLDA (two different regularization parameters were used: η=1 and η=10^{-4}) and 2D-LDA, each class

is composed of 4 images (1 reference + 3 images in different illumination conditions). In terms of classification accuracy, 2D-CCA is better than CCA and is compatible with rLDA and 2D-LDA. However, the running time of 2D-CCA is about 2 sec, whereas it takes about 150 sec for CCA.

Table 2: Averaged accuracy in Experiment 1. The reduced dimension is represented by d and the best case for each method was chosen through experiments.

Method	Accuracy
CCA	92.56% (d=40)
2D-CCA	98.46% (d=40)
rLDA (η= 1)	90.51% (d=40)
rLDA (η=10^{-4})	98.97% (d=40)
2D-LDA	98.21% (d=35)

Similarly to CCA, in MCCA and 2DMCCA experiments, each face image was resized by 25×25 and was normalized according to the eye location because of complexity of data.

We divide the dataset into four groups where one group contains reference images and the other groups contain images with different illumination conditions, i.e, each group $\{\boldsymbol{X}_i^t, t = 1, \cdots, N\}$ consists of 134 face images in different illumination conditions(in this case, N=134). Fig. 2 shows the organization of groups.

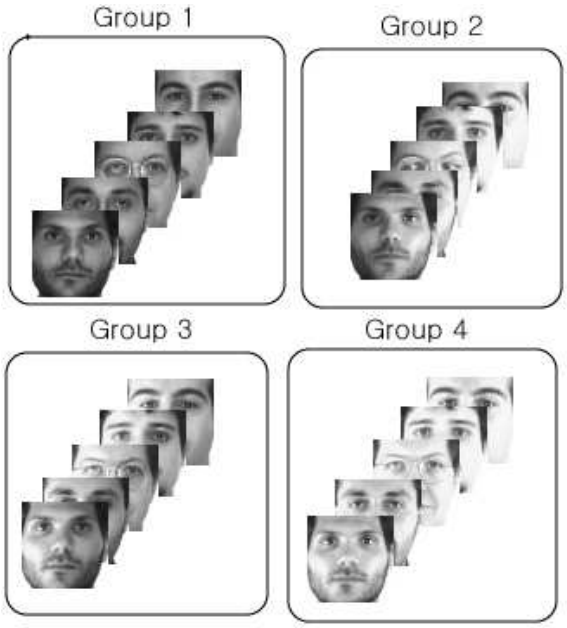

Figure 2: For this experiment, exemplary of orgranzation of face images are shown. Each group contains different illumination conditions. First group contains reference images and the other groups are right, left and both illumination conditions respectively.

For MCCA, images are reshaped into 625-dimensional vectors and d-dimensional canonical correlation variable vectors ($\boldsymbol{W}_i^\top \boldsymbol{x}_i^t$) are used in classification. In the case of 2D-MCCA, images are directly used and $d \times d$ canonical correlation variable matrices ($\boldsymbol{L}_i^\top \boldsymbol{X}_i^t \boldsymbol{R}_i$) are used as inputs to 1-NN classifier. We carry out the experiment with constraint 4 in MCCA

and constraint (26) in 2D-MCCA. With 10-fold cross-validation, the averaged accuracy is summarized in Table 3 and 4.

Table 3: Averaged accuracy in AR Experiment for four groups for MCCA. The reduced dimension is represented by d.

Dimension	10	20	30	40	50
	60	70	80	90	100
Constraint 4	95.13 %	97.18 %	97.69 %	97.95 %	97.69 %
	97.18 %	96.92 %	96.92 %	96.92 %	96.67 %

Table 4: Averaged accuracy in AR Experiment for four groups for 2DMCCA. The reduced dimension is represented by $d \times d$.

Dimension	5×5	10×10	15×15	20×20	25×25
Constraint 1	97.44 %	97.95 %	97.44 %	97.44 %	97.18 %

In terms of classification accuracy, 2D-MCCA is compatible with MCCA. However, the running complexity of 2D-MCCA is much faster than MCCA because 2D methods do not reshape images into long vector.

6.2 Experiment 2: Face images in different poses

We used a subset of FERET face DB which consists of 1800 face images (of size 40×50) of 600 people. For each person, there are 3 images associated with 3 different poses, including front (F), left (L), and right (R). Face images were normalized manually with respect to eye locations. Exemplary images are shown in Fig. 3.

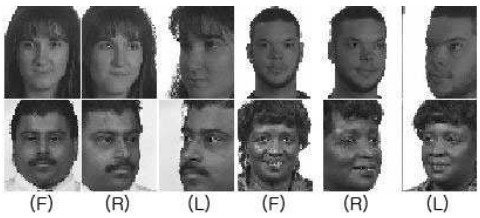

Figure 3: Exemplary face images in FERET face DB.

In order to apply CCA and 2D-CCA, we partition the dataset into 2 groups where $\{\boldsymbol{X}_t\}$ is a set of face images with frontal view and $\{\boldsymbol{Y}_t\}$ is a set of face images with non-frontal but the same pose. We apply CCA and 2D-CCA to a pair of front-left and front-right. We also apply rLDA and 2D-LDA. In such a case, each class contains 1 frontal and 1 non-frontal face image, which might be an ideal case for LDA. As in Experiment 1, averaged accuracy with 10-fold cross-validation is summarized in Table 5. It is expected that LDA methods do not work well since each class contains only 2 images. In contrast, 2D-CCA exploits a linear relationship to identify a corresponding reference (frontal view) image, given a non-frontal face image, showing its reasonable better performance compared to other methods. The running time of 2D-CCA is about 2 sec, whereas CCA takes about 80 sec.

Figs. 4 and 5 show the classification accuracy of CCA and 2D-CCA, with respect to varying d (reduced dimension). In contrast to CCA, 2D-CCA is not significantly influced by

Table 5: Averaged accuracy in Experiment 2.

Method	Front-Left Accuracy	Front-Right Accuracy
CCA	56.17% (d=20)	61.33% (d=10)
2D-CCA	**72.33**% (d=10)	**74.17**% (d=10)
rLDA (η=1)	32.33% (d=40)	34.83% (d=60)
rLDA (η=10^{-4})	53.83% (d=70)	55.50% (d=60)
2D-LDA	29.50% (d=15)	33.33% (d=30)

the size of the reduced dimension. This might result from the fact that 2D-CCA correlates relationships between image datasets with preserving their spatial structure.

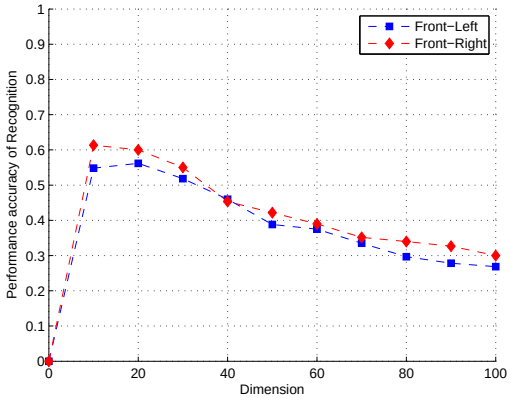

Figure 4: The classification accuracy is shown with respect to the reduced dimension d in CCA.

Similarly to CCA, in MCCA and 2DMCCA experiments, we partition the dataset into 3 groups where the first group $\{\boldsymbol{X}_1^t\}$ is a set of face images with frontal view and the other groups are sets of face images with non-frontal. Fig. 6 shows the organization of groups.

As in Experiment 1, averaged accuracy with 10-fold cross-validation is summarized in Table 6 and 7.

Table 6: Averaged accuracy in experiments 2 with MCCA.

Dimension	10	20	30	40	50
	60	70	80	90	100
Constraint 4	22.96 %	17.83 %	13.58 %	11.17 %	9.51 %
	8.64 %	7.78 %	6.60 %	5.62 %	5.37 %

Table 7: Averaged accuracy in experiment2 with 2D-MCCA.

Dimension	4x5	8x10	12x15	16x20	20x25	24x30
Constraint 1	56.67 %	55.62 %	49.88 %	49.44 %	46.98 %	46.17 %

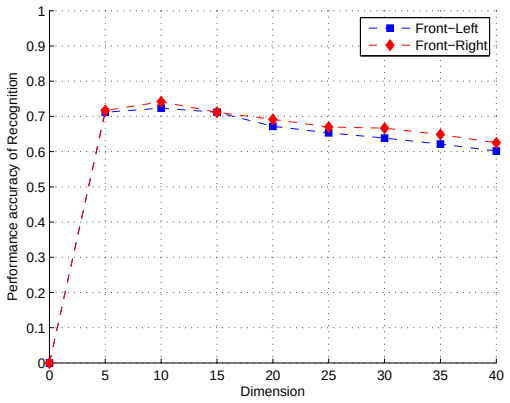

Figure 5: The classification accuracy is shown with respect to the reduced dimension $d \times d$ in 2D-CCA.

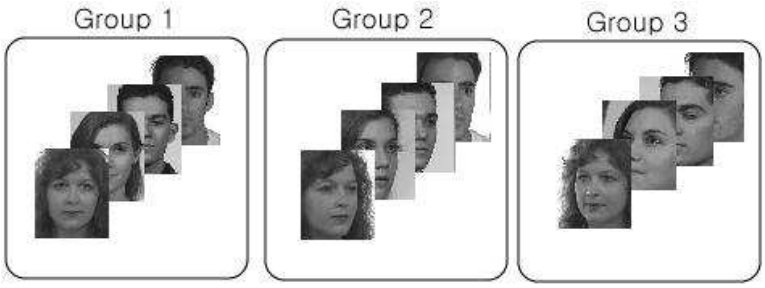

Figure 6: For this experiment, exemplary of orgranzation of face images are shown. Each group contains different poses. First group contains reference images and the other groups are right, left poses respectively.

7 Conclusions

In this report we have presented a two-dimensional extension of CCA and MCCA which directly takes image data as inputs without reshaping them into vectors, in order to correlate relationships between them. The main advantage of 2D-CCA and 2D-MCCA is two-fold: (1) low computational complexity; (2) preserving spatial structure of image data in the calculation of canonical variable matrices. We have also presented novel applications of 2D-CCA, including correspondence learning between face images in different illumination conditions or in different poses.

Appendix: Detailed Derivation

We first solve (9) for $\boldsymbol{l}_x$ and $\boldsymbol{l}_y$, given $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$. Then we solve solve (10) for $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$, given $\boldsymbol{l}_x$ and $\boldsymbol{l}_y$. Repeating these until convergence, leads to the solution to our 2D-CCA. In order to determine $\boldsymbol{l}_x$ and $\boldsymbol{l}_y$ which solves (9), we consider the following Lagrangian

$$\begin{aligned} \mathcal{J}_l &= \boldsymbol{l}_x^\top \boldsymbol{\Sigma}_{xy}^r \boldsymbol{l}_y + \lambda_{lx}\left(1 - \boldsymbol{l}_x^\top \boldsymbol{\Sigma}_{xx}^r \boldsymbol{l}_x\right) \\ &\quad + \lambda_{ly}\left(1 - \boldsymbol{l}_y^\top \boldsymbol{\Sigma}_{yy}^r \boldsymbol{l}_y\right). \end{aligned} \tag{37}$$

Solving $\frac{\partial \mathcal{J}_l}{\partial \boldsymbol{l}_x} = 0$ and $\frac{\partial \mathcal{J}_l}{\partial \boldsymbol{l}_y} = 0$, leads to

$$\boldsymbol{\Sigma}_{xy}^r \boldsymbol{l}_y - 2\lambda_{lx} \boldsymbol{\Sigma}_{xx}^r \boldsymbol{l}_x = 0, \tag{38}$$

$$\boldsymbol{\Sigma}_{yx}^r \boldsymbol{l}_x - 2\lambda_{ly} \boldsymbol{\Sigma}_{yy}^r \boldsymbol{l}_y = 0. \tag{39}$$

Pre-multiplying both sides of (38) by $\boldsymbol{l}_x^\top$ with taking the constraint $\boldsymbol{l}_x^\top \boldsymbol{\Sigma}_{xx}^r \boldsymbol{l}_x = 1$ into account, leads to

$$\boldsymbol{l}_x^\top \boldsymbol{\Sigma}_{xy}^r \boldsymbol{l}_y = 2\lambda_{lx}. \tag{40}$$

Pre-multiplying both sides of (39) by $\boldsymbol{l}_y^\top$ with taking the constraint $\boldsymbol{l}_y^\top \boldsymbol{\Sigma}_{yy}^r \boldsymbol{l}_y = 1$ into account, leads to

$$\boldsymbol{l}_y^\top \boldsymbol{\Sigma}_{yx}^r \boldsymbol{l}_y = 2\lambda_{ly}. \tag{41}$$

It follows from (40) and (41) that $\lambda = 2\lambda_{lx} = 2\lambda_{ly}$. Thus, Eqs. (38) and (39) yields the generalized eigenvalue problem given in (14). Assuming $\boldsymbol{\Sigma}_{yy}^r$ is invertible, it follows from Eq. (39) that we have

$$\boldsymbol{l}_y = \frac{1}{\lambda}\left[\boldsymbol{\Sigma}_{yy}^r\right]^{-1} \boldsymbol{\Sigma}_{yx}^r \boldsymbol{l}_x. \tag{42}$$

Substituting (42) into (38) gives

$$\boldsymbol{\Sigma}_{xy}^r \left[\boldsymbol{\Sigma}_{yy}^r\right]^{-1} \boldsymbol{\Sigma}_{yx}^r \boldsymbol{l}_x = \lambda^2 \boldsymbol{\Sigma}_{xx}^r \boldsymbol{l}_x. \tag{43}$$

Thus, the generalized eigenvectors in (14) are determined by first solving for the generalized eigenvectors of (43) and finding $\boldsymbol{l}_y$ using (42). In a similar manner, $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$ which solves (10) is determined by the generalized eigenvalue problem given in (15).

References

[1] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with applications to learning methods. *Neural Computation*, 16:2639–2664, 2004.

[2] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:312–377, 1936.

[3] J. R. Kettenring. Canonical analysis of several sets of variables. *Biometrika*, 58:433–451, 1971.

[4] P. L. Lai and C. Fyfe. Kernel and nonlinear canonical correlation analysis. In *Proceedings of International Joint Conference on Neural Networks*, 2000.

[5] Y. Li and J. Shawe-Taylor. Using KCCA for Japanese-English cross-language information retrieval and document classification. *Journal of Intelligent Information Systems*, 27(2):117–133, 2006.

[6] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Face recognition using LDA based algorithms. *IEEE Trans. Neural Networks*, 14(1):195–200, 2003.

[7] A. Vinokourov, J. Shawe-Taylor, and J. Cristianini. Inferring a semantic representation of text via cross-language correlation analysis. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.

[8] J. Yang, D. Zhang, A. F. Frangi, and J. Y Yang. Two-dimensional PCA: A new approach to appearance-based face representation and recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(1):131–137, 2004.

[9] J. Ye, R. Janardan, and Q. Li. GPCA: An efficient dimension reduction scheme for image compression and retrieval. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 354–363, Seattle, WA, 2004.

[10] J. Ye, R. Janardan, and Q. Li. Two-dimensional linear discriminant analysis. In *Advances in Neural Information Processing Systems*, pages 1569–1576. MIT Press, 2005.

[11] W. Zheng, X. Zhou, C. Zou, and L. Zhao. Facial expression recognition using kernel canonical correlation analysis (KCCA). *IEEE Trans. Neural Networks*, 17(1):223–238, 2006.

{sunholee,seungjin}@postech.ac.kr


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교


M L G
POSTECH