Supervised Local Subspace Learning for Continuous Head Pose Estimation

Dong Huang[1 3] Markus Storer[2] Fernando De la Torre[1] Horst Bischof[2]
[1]Robotics Institute, Carnegie Mellon University, USA
[2]Institute for Computer Graphics and Vision, Graz University of Technology, Austria
[3]University of Electronic Science and Technology of China, China
dghuang@andrew.cmu.edu, storer@icg.tugraz.at, ftorre@cs.cmu.edu, bischof@icg.tugraz.at

Abstract

Head pose estimation from images has recently attracted much attention in computer vision due to its diverse applications in face recognition, driver monitoring and human computer interaction. Most successful approaches to head pose estimation formulate the problem as a nonlinear regression between image features and continuous $3D$ angles (i.e. yaw, pitch and roll). However, regression-like methods suffer from three main drawbacks: (1) They typically lack generalization and overfit when trained using a few samples. (2) They fail to get reliable estimates over some regions of the output space (angles) when the training set is not uniformly sampled. For instance, if the training data contains under-sampled areas for some angles. (3) They are not robust to image noise or occlusion. To address these problems, this paper presents Supervised Local Subspace Learning (SL^2), a method that learns a local linear model from a sparse and non-uniformly sampled training set. SL^2 learns a mixture of local tangent spaces that is robust to under-sampled regions, and due to its regularization properties it is also robust to over-fitting. Moreover, because SL^2 is a generative model, it can deal with image noise. Experimental results on the CMU Multi-PIE and BU-3DFE database show the effectiveness of our approach in terms of accuracy and computational complexity.

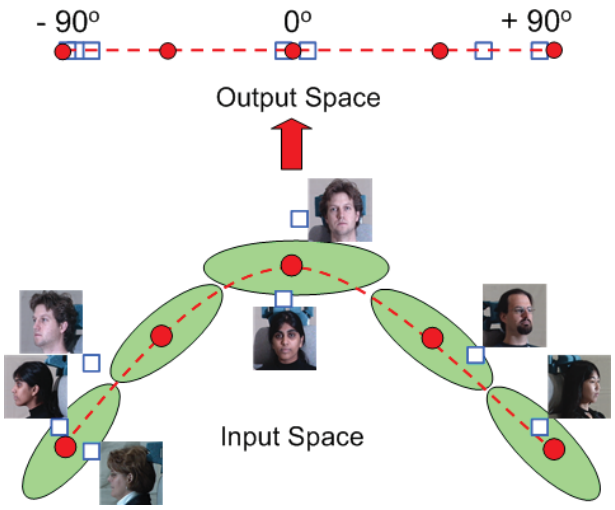


Figure 1. Illustration of the Supervised Local Subspace Learning (SL^2) algorithm for continuous head pose estimation. Blue squares represent training samples, and red dots and green ellipses the local mean and covariance of the tangent spaces respectively, learned by SL^2. Observe that the training data is sparse and it does not uniformly sample the space of yaw angle.

1. Introduction

Automatic estimation of head pose from images has been a hot research topic over the last decade, inspired by the increasing demands of many applications such as human-computer interaction, human behavior understanding and driver monitoring. For instance, in human-computer interaction computers are required to determine the gaze as well as interpreting head gesturing, e.g., the meaning of head movements like nodding. This enables very direct means to interact with virtual worlds, especially for computer gaming. Langton et al. [14] showed that the gaze direction is a combination of head pose and eye gaze. Head pose estimation is also essential to determine the eye-gaze viewing direction of a person for applications in driver monitoring or social interaction. In the area of automotive safety, the driver's head has been monitored to recognize driver distraction and inattention to avoid vehicle collisions [21]. Face recognition algorithms use head pose estimation to render frontal views to improve recognition accuracy [4].

Head pose estimation from images is a challenging problem because of the large variability of image changes due to pose, expression, illumination, facial expressions, and subject variability. A generic (i.e. person-independent) algorithm for head pose estimation has to be robust to such biological factors and also robust to occlusion, noise, lighting and perspective distortion. There exists a large amount of

literature on the topic of head pose estimation, see [22] for a review. Broadly speaking, these methods can be divided into: template based methods [23, 3], classification based methods [12, 13], regression based methods [16, 18, 20], embedding based methods [26, 25, 2, 1], Active Appearance Models (AAMs) [6] and geometric methods [10]. Template and classification based methods are based on template comparisons that assign test facial images to discrete angles. Typically these methods might suffer from over-fitting and are highly sensitive to non-uniform sampled training data. Manifold embedding techniques produce a low dimensional representation of the original facial features in supervised or unsupervised fashion and then uses template, classification or regression methods to learn a mapping from the low dimensional manifold to the angles. Generative approaches such as AAMs fit a shape and appearance models to the image, typically suffering from lack of generalization to previously unseen subjects and local minima problems during the fitting. The geometric methods rely on the detection of facial features like corners of nose or mouth and wrongly detected features can degrade the overall performance significantly. One of the major challenges in learning-based methods for head pose estimation is to gather enough uniformly distributed training data for the three angles. Furthermore, classification and regression-based methods are not robust to image noise or occlusions in the testing data, whereas generative models such as AMMs or template based methods can be easily robustified.

To address the limitations of previous methods, this paper proposes Supervised Local Subspace Learning (SL^2), a method that builds local linear models from a sparse and non-uniformly sampled training set. SL^2 learns a mixture of local tangent spaces that is robust to under-sampled regions and due to its regularization properties, robust to overfitting. Moreover, because it is a generative model, can be easily robustified to deal with image noise. Figure 1 illustrates the main idea behind SL^2 and its applications to pose estimation. Given a very sparse set of training images (blue squares in the input space) and the corresponding angles (blue squares) in the output space, SL^2 learns a set of local subspaces parameterized by a center (red circle) and a tangent subspace (green ellipses). Experiments in two publicly available databases (BU-3DFE [28] and CMU Multi-PIE [11]) illustrate the benefits of our approach in comparison to state-of-the-art methods.

2. Related Work

This section reviews existing methods for continuous head pose estimation and discusses its relation to our work. For a more extensive survey on head pose estimation problems can be found in [22].

Appearance template methods [23, 3] estimate head pose by directly comparing facial images with a set of template images, and assign the angle of the most similar template. Template based comparison methods such as K-Nearest-Neighbor (KNN), are usually sensitive to noise (or undesirable factors such as expression or illumination) and typically need uniformly sampled training data to get accurate results.

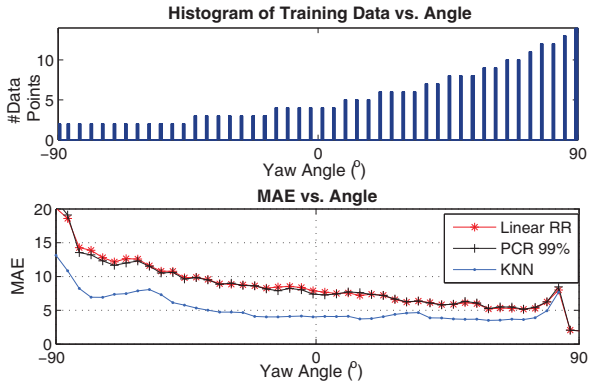
Classification based methods [12, 13] learn a mapping between the input image and a discretized space of poses. There are a number of classifiers that have been used such as multiclass SVMs [15], multiclass linear discriminant analysis (LDA) or the kernelized version (KLDA [8]). Given a new test image, the classifiers assign it to a discrete pose class [27]. These methods also suffer from the non-uniform sampling in the training data and only return discrete head pose estimates. Moreover, they are particularly sensitive to noise in the data (training and testing).

Regression based methods estimate the pose by learning a linear or nonlinear function between the image and continuous angles. Several regressors are possible such as (SVRs) [16, 18, 20] and Gaussian Processes Regression (GPR) [24]. As classification methods, regression based methods suffer from irregular distributed training data, overfit in presence of limited training samples and are not robust to noise in the training or testing images.

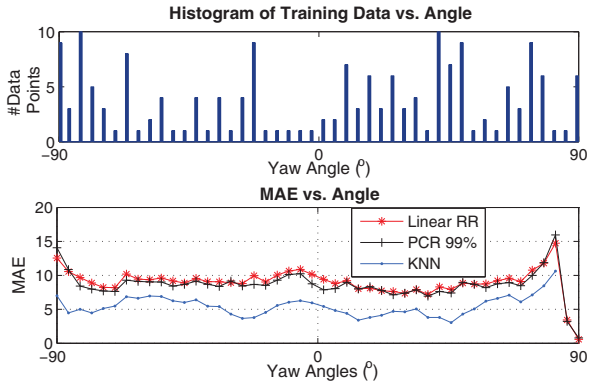
Embedding based methods assume that the pose variation can be represented by a low-dimensional space [19]. After learning the embedding, a new test image is embedded into the low dimensional manifolds and then classification or regression methods are used to compute the angle in a supervised manner. However, most embedding based methods are unsupervised in nature and do not extract features that incorporate class information. Therefore, there is no guarantee that the variations in poses can dominate over other appearance variations such as identity, age, expression and lighting. To partially solve this problem, Balasubramanian et al. [1] used metric distance learning to remove irrelevant dimensions not related to pose changes. This strategy has shown to improve head pose estimation performance using Isomap [26, 1], locally linear embedding (LLE) [25], and Laplacian Eigenmaps (LE) [2]. However, embedded methods still rely on template matching or regression in the embedded space [9] for the final angle estimation. Moreover, finding the optimal dimension is still an open research problem.

3. Supervised Local Subspace Learning (SL^2)

This section describes in detail the motivation, formulation and the algorithm for SL^2.



(a) Sampling with a monotonically increasing number of data points.



(b) Sampling with a random number of data points.

Figure 2. Yaw angle prediction from images in the FacePix database [17] as a function of training samples. Three methods are compared: linear RR, PCR (preserving 99% of the energy), KNN (k=4) using two different sampling strategies. (a) Sampling with a monotonically changing number of samples. The 90° angle has more samples than the -90°. (b) Randomly Gaussian sampling over angles. All training samples are selected randomly. Both sampling strategies are repeated 50 times and the mean absolute angular error (MAE) is displayed.

3.1. Motivation

This section illustrates some issues of current algorithms for pose estimation. Let $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \in \Re^{d \times n}$ [1] be a matrix that contains the d-dimensional HOG features [7] of n facial images and let $\mathbf{\Theta} = \{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_n\} \in \Re^{3 \times n}$ be, in general, the 3-dimensional angles (i.e., the yaw, pitch and roll angle) representing the head's pose.

Linear regression methods for pose estimation learn a transformation $\mathbf{A} \in \Re^{3 \times d}$ between image features $\mathbf{X}$ and angles $\mathbf{\Theta}$ by minimizing: $\mathbf{A} = \arg\min_{\mathbf{A}} \|\mathbf{\Theta} - \mathbf{A}\mathbf{X}\|_F^2$. The optimal $\mathbf{A}$ is $\mathbf{A} = \mathbf{\Theta}\mathbf{X}^T \left(\mathbf{X}\mathbf{X}^T\right)^{-1}$. However, the matrix $\mathbf{X}\mathbf{X}^T \in \Re^{d \times d}$ is typically rank-deficient due to the high-dimensionality of the data. Standard methods to solve the small sample size problems include Ridge Regression (RR) that introduces a regularization term, or Principal Component Regression (PCR) that does the regression on the principal components of the data. PCR can potentially remove dimensions that are maximally correlated with the angles and RR typically bias the solution towards small angles. On the other hand, the parameters in regression approaches are computed as averages over training samples, hence representing more poorly the angles that are under-sampled. This effect is illustrated in Figure 2 that shows the pose prediction error in the FacePix database [17]. Figure 2 (a) and (b) illustrates the errors for three methods using two different sampling strategies (see caption). In both strategies training data is sampled randomly and the rest is used for testing. For both sampling strategies the experiments are repeated 50 times and the mean absolute angular error (MAE) is displayed. As shown in Figure 2, regression methods (linear RR and PCR) tend to produce larger errors in the under-sampled angle regions, whereas k-nearest neighbor (KNN) steadily outperforms regression methods. This is because KNN estimates the angles as locally weighted combinations of neighboring samples, being less sensitive to the distribution of training data.

In the case of sparse training samples, KNN has shown better performance than regression methods. However, a major disadvantage of KNN during testing is its computational complexity. KNN needs to compute the similarity between the test sample with all training data, and this can be time consuming for many applications. Alternatively, non-linear regression methods (e.g. kernel regression) suffer from a similar weakness as linear regression and KNN approaches: (a) parameters of regression can be biased towards densely sampled areas; (b) computationally expensive to match a test point against all training data; (c) lack of generalization when using few training samples; (d) lack of robustness to noise. To partially solve these issues, we propose Supervised Local Subspace Learning (SL^2).

3.2. Parametrization of local subspaces

Image features in $\mathbf{X}$ are often very high dimensional and typically are non-linearly related to the angles $\mathbf{\Theta}$. The basic idea of our method is to model image features $\mathbf{X}$ with m local subspaces parameterized by the angles $\mathbf{\Theta}$, that is:

$$\mathbf{X} = f(\mathbf{\Theta}) \approx f\left(\langle \mathbf{c}_1, \mathbf{G}_1, \widehat{\boldsymbol{\theta}}_1 \rangle, \cdots, \langle \mathbf{c}_m, \mathbf{G}_m, \widehat{\boldsymbol{\theta}}_m \rangle, \mathbf{\Theta}\right). \quad (1)$$

where $\langle \mathbf{c}_i, \mathbf{G}_i, \widehat{\boldsymbol{\theta}}_i \rangle$ refers to the i^{th} local subspace, $\mathbf{c}_i \in \Re^d$ is the mean and $\mathbf{G}_i \in \Re^{d \times 3}$ a basis for the tangent space.

[1] Bold capital letters denote matrices $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$. $\mathbf{x}_j$ represents the j^{th} column of the matrix $\mathbf{X}$. All non-bold letters represent scalar variables. x_{ij} denotes the scalar in the row i and column j of the matrix $\mathbf{X}$ and the scalar i-th element of a column vector $\mathbf{x}_j$. $\mathbf{I}_k \in \Re^{k \times k}$ denotes the identity matrix. $\|\mathbf{x}\|_2$ denotes the L2-norm of the vector $\mathbf{x}$. $tr(\mathbf{A}) = \sum_i a_{ii}$ is the trace of the matrix $\mathbf{A}$. $\|\mathbf{A}\|_F^2 = tr(\mathbf{A}^T\mathbf{A}) = tr(\mathbf{A}\mathbf{A}^T)$ designates the Frobenious norm of matrix $\mathbf{A}$.

Observe that the latent variable (the angle $\widehat{\boldsymbol{\theta}}$) changes continuously and it is bounded.

To deal with the problem of non-uniform sampling of the training set, we uniformly sample the output space (angle) generating several new angles $\widehat{\boldsymbol{\theta}}_i$, such that the space of angles is uniformly sampled. However, for these new angles, $\widehat{\boldsymbol{\theta}}_i$, the image features in the input space are unknown, and hence cannot be used for testing. For each new sampled angles, $\widehat{\boldsymbol{\theta}}_i$, we will associate a subspace parameterized by a mean $\mathbf{c}_i$ and a basis $\mathbf{G}_i$.

We approximate the function $f(\cdot)$ in Eqn. (1) by its first order Taylor expansion in the center $\mathbf{c}_i$ of the i^{th} subspace (there are m local subspaces). Each training data point $\mathbf{x}_p, \boldsymbol{\theta}_p$, will be reconstructed from a subset of local subspaces. These local subspaces will be determined by the proximity in angles. That is, the subset of subspaces from which $\mathbf{x}_p$ will be reconstructed belong to the neighborhood of the angles close to $\boldsymbol{\theta}_p$, mathematically $\Gamma_p = \{i | \widehat{\boldsymbol{\theta}}_i \in \mathcal{N}(\boldsymbol{\theta}_p), i = 1, \cdots, m\}$. $\mathcal{N}(\boldsymbol{\theta}_p)$ denotes the index to the neighborhood of $\boldsymbol{\theta}_p$. Then, $\mathbf{x}_p$ is approximated as:

$$\mathbf{x}_p \approx \mathbf{c}_i + \mathbf{G}_i \Delta\boldsymbol{\theta}_{pi} \tag{2}$$

where $\Delta\boldsymbol{\theta}_{pi} = \boldsymbol{\theta}_p - \widehat{\boldsymbol{\theta}}_i$. Observe that $\boldsymbol{\theta}_p$ are the set of angles that are provided in a supervised manner, and $\widehat{\boldsymbol{\theta}}_i$ are a set of angles that will be uniformly resampled. Eqn. (2) is used as the building block of the error function of SL^2 described in the next section.

3.3. Error function for SL^2

SL^2 minimizes:

$$\begin{aligned} E\left(\mathbf{c}_i, \mathbf{G}_i\right) &= \sum_{p=1}^{n} \sum_{i \in \Gamma_p} w_{pi}^2 \|\mathbf{x}_p - (\mathbf{c}_i + \mathbf{G}_i \Delta\boldsymbol{\theta}_{pi})\|_2^2 \\ &+ \lambda \sum_{j=1}^{m} \sum_{i \in \Gamma_j} w_{ji}^2 \|\mathbf{c}_j - (\mathbf{c}_i + \mathbf{G}_i \Delta\boldsymbol{\theta}_{ji})\|_2^2, \end{aligned} \tag{3}$$

Eqn. (3) has two terms. In the first term, each training sample $(\mathbf{x}_p, \boldsymbol{\theta}_p)$ is approximated independently using a subset of local subspaces. The local subspaces are selected by the angles that are close to $\boldsymbol{\theta}_p$. The second factor, is a regularization term, that enforces that the mean of each subspace is to be reconstructed by the neighboring subspaces. This term ensures that the subspace parameters vary smoothly and can be estimated from sparse non-uniform data. The parameter $\lambda = (n/m)^2$ balances the importance of both terms. The parameter w_{pi}, weights the contribution of each neighboring subspace to the data sample reconstruction, and it is computed as:

$$w_{pi} = \frac{\psi(\boldsymbol{\theta}_p, \widehat{\boldsymbol{\theta}}_i)}{\sum_{q=1}^{|\Gamma_p|} \psi(\boldsymbol{\theta}_q, \widehat{\boldsymbol{\theta}}_i)}, \tag{4}$$

where the function $\psi(\cdot)$ in Eqn. (4) can be any positive valued function inversely proportional to the distance between angles. For simplicity, in the following, we selected the following function $\psi(\boldsymbol{\theta}_p, \widehat{\boldsymbol{\theta}}_i) = \frac{1}{\|\boldsymbol{\theta}_p - \widehat{\boldsymbol{\theta}}_i\|_2^2}$. w_{ji} is defined similarly.

3.4. Learning C and G

Let $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_m] \in \Re^{d \times m}$ be a matrix that contains the mean for the local subspaces and $\mathbf{G} = [\mathbf{G}_1, \cdots, \mathbf{G}_m] \in \Re^{d \times 3m}$ be the concatenated basis associated with these subspaces. We used an Alternated Least-Squares (ALS) algorithm to optimize for $\mathbf{C}$ and $\mathbf{G}$. The ALS alternates between fixing $\mathbf{C}$ and estimate $\mathbf{G}$ (a linear problem) and fixing $\mathbf{G}$ while estimating $\mathbf{C}$ (a linear problem). It is easy to show that Eqn. (3) is bounded below, and the ALS will monotonically decrease the error. ALS methods typically converge fast at the beginning and slower close to the critical point. Without losing generality, we initialize the i^{th} mean $\mathbf{c}_i$ as the average of the nearest training data points of $\mathbf{x}_p$ where $p = \arg\min_p \|\widehat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_p\|_2^2$.

3.4.1 Optimizing over G

$E(\mathbf{C}, \mathbf{G})$ can be rewritten as follows for a fixed $\mathbf{C}$:

$$\begin{aligned} E_1\left(\mathbf{G}\right) &= \sum_{p=1}^{n} \left\| \left[\mathbf{x}_p \mathbf{e}^T - \left(\mathbf{C} + \mathbf{G}\Delta\boldsymbol{\Theta}_p\right) \mathbf{S}_{\Gamma_p}\right] \mathbf{W}_p \right\|_F^2 \\ &+ \lambda \sum_{j=1}^{m} \left\| \left[\mathbf{c}_j \mathbf{e}^T - \left(\mathbf{C} + \mathbf{G}\Delta\boldsymbol{\Theta}_j\right) \mathbf{S}_{\Gamma_j}\right] \mathbf{W}_j \right\|_F^2, \end{aligned}$$

where

$$\Delta\boldsymbol{\Theta}_p = \begin{pmatrix} \Delta\boldsymbol{\theta}_{p1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Delta\boldsymbol{\theta}_{p2} & \cdots & \mathbf{0} \\ \cdots & \cdots & \ddots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Delta\boldsymbol{\theta}_{pm} \end{pmatrix} \in \Re^{3m \times m},$$

and $\mathbf{S}_{\Gamma_p} \in \Re^{m \times k}$ is a $0-1$ selection matrix. The diagonal weighting matrices $\mathbf{W}_p = diag\{w_{pi}, i \in \Gamma_p\}$ and $\mathbf{W}_j = diag\{w_{ji}, i \in \Gamma_j\}$ (as Eqn. (4)). Minimizing $E_1\left(\mathbf{G}\right)$ w.r.t. $\mathbf{G}$ leads to:

$$\mathbf{G} = (\mathbf{A}_{\mathbf{G}}^{(\mathbf{x})} + \mathbf{A}_{\mathbf{G}}^{(\mathbf{c})})(\mathbf{B}_{\mathbf{G}}^{(\mathbf{x})} + \mathbf{B}_{\mathbf{G}}^{(\mathbf{c})})^{-1}, \tag{5}$$

where $\mathbf{A}_{\mathbf{G}}^{(\mathbf{x})} = \sum_{p=1}^{n} \left(\mathbf{x}_p \mathbf{e}^T - \mathbf{C}\mathbf{S}_{\Gamma_p}\right) \mathbf{W}_p \mathbf{W}_p^T \mathbf{S}_{\Gamma_p}^T \Delta\boldsymbol{\Theta}_p^T$, and $\mathbf{A}_{\mathbf{G}}^{(\mathbf{c})} = \lambda \sum_{j=1}^{m} \left(\mathbf{c}_j \mathbf{e}^T - \mathbf{C}\mathbf{S}_{\Gamma_j}\right) \mathbf{W}_j \mathbf{W}_j^T \mathbf{S}_{\Gamma_j}^T \Delta\boldsymbol{\Theta}_j^T$. Similarly, $\mathbf{B}_{\mathbf{G}}^{(\mathbf{x})} = \sum_{p=1}^{n} \Delta\boldsymbol{\Theta}_p \mathbf{S}_{\Gamma_p} \mathbf{W}_p \mathbf{W}_p^T \mathbf{S}_{\Gamma_p}^T \Delta\boldsymbol{\Theta}_p^T$, and $\mathbf{B}_{\mathbf{G}}^{(\mathbf{c})} = \lambda \sum_{j=1}^{m} \Delta\boldsymbol{\Theta}_j \mathbf{S}_{\Gamma_j} \mathbf{W}_j \mathbf{W}_j^T \mathbf{S}_{\Gamma_j}^T \Delta\boldsymbol{\Theta}_j^T$.

3.4.2 Optimizing over C

Similarly, we can rewrite $E(\mathbf{C},\mathbf{G})$ for a fixed $\mathbf{G}$ as:

$$E_2(\mathbf{C}) = \sum_{p=1}^{n} \left\| \left[\mathbf{x}_p \mathbf{e}^T - (\mathbf{C} + \mathbf{G}\Delta\boldsymbol{\Theta}_p)\mathbf{S}_{\Gamma_p} \right] \mathbf{W}_p \right\|_F^2 + \lambda \sum_{j=1}^{m} \left\| \left[\mathbf{C}\left(\mathbf{s}_j\mathbf{e}^T - \mathbf{S}_{\Gamma_j}\right) - \mathbf{G}\Delta\boldsymbol{\Theta}_j\mathbf{S}_{\Gamma_j} \right] \mathbf{W}_j \right\|_F^2$$

where $\mathbf{s}_j$ is the $0-1$ selection vector such that $\mathbf{C}\mathbf{s}_j = \mathbf{c}_j$, and

$$\Delta\boldsymbol{\Theta}_j = \begin{pmatrix} \Delta\boldsymbol{\theta}_{j1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Delta\boldsymbol{\theta}_{j2} & \cdots & \mathbf{0} \\ \cdots & \cdots & \ddots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Delta\boldsymbol{\theta}_{jm} \end{pmatrix} \in \Re^{3m\times m}.$$

Optimizing $E(\mathbf{C},\mathbf{G})$ over $\mathbf{G}$ results in,

$$\mathbf{C} = (\mathbf{A}_{\mathbf{C}}^{(\mathbf{x})} + \mathbf{A}_{\mathbf{C}}^{(\mathbf{c})})(\mathbf{B}_{\mathbf{C}}^{(\mathbf{x})} + \mathbf{B}_{\mathbf{C}}^{(\mathbf{c})})^{-1}, \tag{6}$$

where $\mathbf{A}_{\mathbf{C}}^{(\mathbf{x})} = \sum_{p=1}^{n} \left(\mathbf{x}_p\mathbf{e}^T - \mathbf{G}\Delta\boldsymbol{\Theta}_p\mathbf{S}_{\Gamma_p}\right)\mathbf{W}_p\mathbf{W}_p^T\mathbf{S}_{\Gamma_p}^T$ and $\mathbf{A}_{\mathbf{C}}^{(\mathbf{c})} = \lambda\sum_{j=1}^{m}\mathbf{G}\Delta\boldsymbol{\Theta}_j\mathbf{S}_{\Gamma_j}\mathbf{W}_j\mathbf{W}_j^T\left(\mathbf{s}_j\mathbf{e}^T - \mathbf{S}_{\Gamma_j}\right)^T$. Similarly, $\mathbf{B}_{\mathbf{C}}^{(\mathbf{x})} = \sum_{p=1}^{n}\mathbf{S}_{\Gamma_p}\mathbf{W}_p\mathbf{W}_p^T\mathbf{S}_{\Gamma_p}^T$, and $\mathbf{B}_{\mathbf{C}}^{(\mathbf{c})} = \lambda\sum_{j=1}^{m}\left(\mathbf{s}_j\mathbf{e}^T - \mathbf{S}_{\Gamma_j}\right)\mathbf{W}_j\mathbf{W}_j^T\left(\mathbf{s}_j\mathbf{e}^T - \mathbf{S}_{\Gamma_j}\right)^T$.

3.5. Estimating the angle for a test data point

For a new test data point $\mathbf{x}_t$, its neighboring subspaces in Γ_t are found in two steps for efficiency purposes: (1) find $2|\Gamma_t|$ candidate subspaces whose centers are closest to x_t in the input space; (2) select the $|\Gamma_t|$ neighboring subspaces from the $2|\Gamma_t|$ candidates with the least reconstruction errors. Then the weight w_{ti} is computed using Eqn. (4) where $\boldsymbol{\theta}_i$ is the angle associated with the center of the i^{th} subspace ($i \in \Gamma_t$). Finally, the optimal angle $\boldsymbol{\theta}_t$ is computed by minimizing the reconstruction error for $\mathbf{x}_t$,

$$E_t(\boldsymbol{\theta}_t) = \sum_{i\in\Gamma_t} w_{ti}^2 \left\| \mathbf{x}_t - \left[\mathbf{c}_i + \mathbf{G}_i(\boldsymbol{\theta}_t - \widehat{\boldsymbol{\theta}}_i) \right] \right\|_2^2 \tag{7}$$

Minimizing $E_t(\boldsymbol{\theta}_t)$ with respect to $\boldsymbol{\theta}_t$ leads to:

$$\boldsymbol{\theta}_t = \left(\sum_{i\in\Gamma_t} w_{ti}^2 \mathbf{G}_i^T\mathbf{G}_i \right)^{-1} \sum_{i\in\Gamma_t} w_{ti}^2 \mathbf{G}_i^T \left(\mathbf{x}_t - \mathbf{c}_i + \mathbf{G}_i\widehat{\boldsymbol{\theta}}_i \right).$$

Observe that during testing SL^2 is more efficient than WKNN, GPR, kernel SVR and Biased manifold. The most computational expensive part of the SL^2 is computing $\boldsymbol{\theta}_t$ (after Eqn. 7), that has a complexity of $O(md+3dk)$, where k is the number of neighboring subspaces, m is number of subspaces and d is the dimension of HOG features. Other methods need to store all training data points, and compute similarities between a test data point and the training set, that has a computational cost about $O(3n+nd)$, where n is the number of training samples, and typically $n \gg m > k$.

4. Experimental results

We provide qualitative and quantitative evaluations of our method and compare it with state-of-the-art approaches on two standard databases: CMU Multi-PIE [11] and BU-3DFE databases [28].

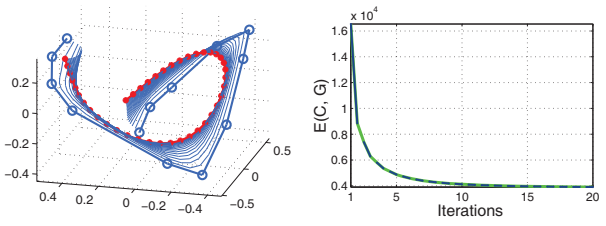
4.1. CMU Multi-PIE

This experiment uses 5648 annotated images from the Multi-PIE database [11], which contains head images of 336 subjects illuminated by a frontal light source under 13 viewing angles and 6 expressions (neutral, smile, surprise, squint, disgust and scream). The yaw angles vary from the left profile ($-90°$) to the right profile ($+90°$), and there is a total of 13 discrete angles spaced by $15°$. All images are cropped according to manually annotated landmark points and normalized to 128×128 pixels. Figure 3 illustrates some examples. We randomly selected 50% of the data for training (2824 images) and the remaining 50% for testing. The subjects in the training are not present during the testing.

Figure 3. Examples of the cropped head images from the CMU Multi-PIE database.

Figure 4 visualizes the training process for the SL^2 with the number of subspaces and the neighboring subspaces being set to 41 and 4 respectively. Figure 4 (a) shows the projection of $\mathbf{c}_i$ onto the first three principal components of the data. Each blue curve represents the set of means ($\mathbf{c}_i$) at one iteration step. At the initial state, (thick blue dots) the means of the subspaces are the image features with the closest angle. Therefore, as shown in Figure 4 (a), the initial 41 subspaces are not equidistantly distributed, many of them overlap with each other (hence only a few blue dots can be seen as compared to red dots). This is because the images in the Multi-PIE database only contains discrete poses separated by $15°$ in yaw angles, and therefore some initial mean assignments are the same. Using SL^2, the means gradually converge to a stable value (thick red dots) where the 41 subspaces are equidistantly spaced. Figure 4 (b) shows how the energy function $E(\mathbf{C},\mathbf{G})$ monotonously decreases using the ALS strategy.

Table 1 presents a quantitative evaluation of SL^2 for different number of subspaces and neighboring subspaces. The number of subspaces are 21, 41, 81 and 161, and the the corresponding spacing between angles is $9°$, $4.5°$, $2.25°$ and $1.125°$ respectively. Table 1 shows that the Mean Absolute Angular Error (MAE) error decreases with the number of subspaces (as expected) but not necessarily with the



(a) (b)

Figure 4. Visualization of the training process for SL^2 (41 subspaces) on the Multi-PIE database. (a) Changes of the subspaces (projection onto the first three principal components) through iterations from the initial state (thick blue dots) to the converged state (thick red dots). (b) $E(\mathbf{C}, \mathbf{G})$ over iterations.

number of neighboring subspaces (unless many subspaces are selected).

Table 1. Quantitative evaluation on the Multi-PIE database. The MAE varies with the number of subspaces ("#S")and the neighboring subspaces ("#NS").

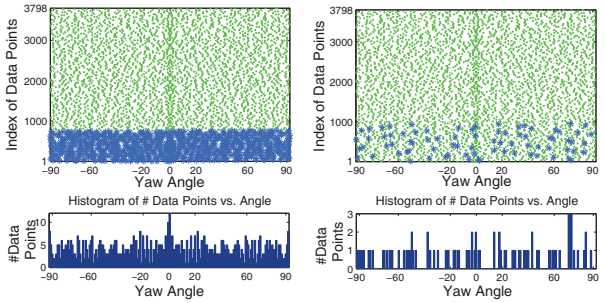
#S \ #NS	2	4	8
21	6.06°	9.18°	12.81°
41	4.60°	5.28°	7.95°
81	4.41°	4.42°	4.86°
161	4.90°	4.56°	4.33°

4.2. Comparison with state-of-the-art methods

This experiment compares the performance of SL^2 with weighted k-nearest neighbor (WKNN) [9], regression based methods such as Linear Ridge Regression (Linear RR), Gaussian Process Regression (GPR) [24] and and Support Vector Regression (SVR) [18]), and Biased Manifold [1]). We used the publicly available BU-3DFE Database [28].

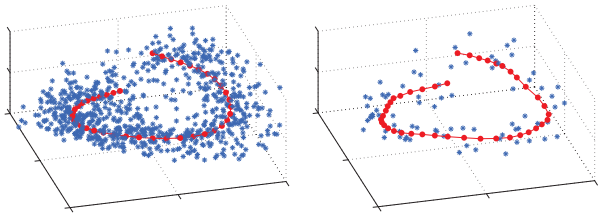
Figure 5. Examples of head images generated from the BU-3DFE database.

To test the performance of the algorithms against non-uniform sampling of the training set, we generated head images by rendering 3D models (laser scans of real human heads) originating from the BU-3DFE database [28]. The 3D database contains 100 subjects (56% female, 44% male), the age ranging from 18 to 70 years old, with a variety of ethnic/racial ancestries, including White, Black, East-Asian, Middle-east Asian, Indian, and Hispanic Latino. The



(a) (b)

Figure 6. Angular distribution of dataset **(I)** consisting of 3798 images generated from the BU-3DFE database with varying yaw angle within $\pm 90°$. The blue stars represent the training data and the green dots represent the testing data. (a) The training set is densely sampled in $\pm 90°$. We used 760 images (about 20%) for training and the remaining 3038 images for testing. (b) The training set is sparsely sampled containing only 79 (about 2.1%) images, the remaining 3719 images are used for testing. The blue starts are the training data.



(a) (b)

Figure 7. Visualization of SL^2 results on dataset **(I)**. Trained means for (a) densely sampled dataset **(I)** and (b) sparsely sampled dataset **(I)**. The dots represents projection onto the three principal components of the data.

textures are provided as real photographs including the information of how to map the texture to the 3D shape of the laser scan. The background of each rendered image was set to an arbitrary image in order to simulate a non-uniform background. We randomly rotated the head models and produced two datasets: **(I)** yaw angle variations within $\pm 90°$ (3798 images); **(II)** pitch and yaw angles within $\pm 30°$ and $\pm 90°$ respectively (6000 images). Some of the generated images can be seen in Figure 5. In both datasets **(I)** and **(II)**, the ground truth of the angles was automatically assigned during rendering without doing a cumbersome manual labeling.

As shown in Figure 6, for dataset **(I)**, we split the training set (blue stars) and testing set (green dots) with to different sampling strategies. First, the training data is densely sampled between $[-90°, 90°]$, and we used 760 images (about 20%) for training and the remaining 3038 images for testing. Second, the training data is sampled very sparsely, containing only 79 (about 2.1%) images, and the remaining 3719 images are used for testing. In both settings, there is no overlap between subjects in training and testing.

Table 2. Comparison of the algorithms on dataset **(I)** (Figure 6) and dataset **(II)** (Figure 8) in terms of the Mean Absolute Error (MAE), Standard Deviation (SD) and the number of data points (NData) required for testing.

MAE±SD (NData)	WKNN	Linear RR	GPR	SVR	Biased Manifold	SL^2
Dataset **(I)** dense	$8.78^\circ \pm 12.92^\circ$ (760)	$22.69^\circ \pm 20.21^\circ$ (1)	$20.89^\circ \pm 18.26^\circ$ (760)	$18.25^\circ \pm 16.42^\circ$ (760)	$8.56^\circ \pm 11.39^\circ$ (760 × 2)	$\mathbf{8.78^\circ \pm 11.67^\circ}$ (41)
Dataset **(I)** sparse	$11.09^\circ \pm 13.37^\circ$ (79)	$29.51^\circ \pm 24.50^\circ$ (1)	$22.72^\circ \pm 20.76^\circ$ (79)	$19.55^\circ \pm 17.55^\circ$ (79)	$9.47^\circ \pm 12.47^\circ$ (79 × 2)	$\mathbf{7.36^\circ \pm 13.11^\circ}$ (41)
Dataset **(II)**	$10.93^\circ \pm 4.57^\circ$ (4188)	$13.86^\circ \pm 8.78^\circ$ (1)	$7.96^\circ \pm 4.54^\circ$ (4188)	$11.80^\circ \pm 7.50^\circ$ (3114)	$15.55^\circ \pm 6.45^\circ$ (4188 × 2)	$\mathbf{5.68 \pm 3.47^\circ}^\circ$ (300)

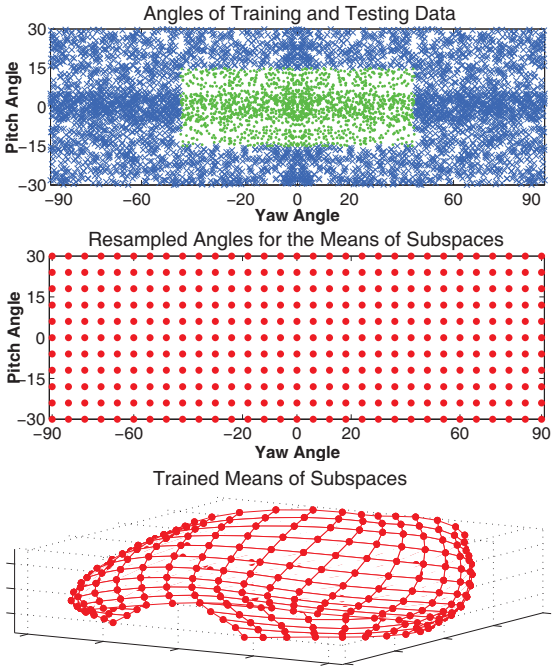


Figure 8. Experiment on dataset **(II)** with SL^2 performing on 6000 images generated from the BU-3DFE database with yaw angles in the range of $\pm 90^\circ$ and pitch angles vary in $\pm 30^\circ$. (First row) The blue crosses represent the 4188 training data and the green dots the 1812 testing data. (Second row) In the output space of the angles, the red dots represent the equidistantly resampled angles for training the subspaces. (Third row) Visualization of the trained means in the input space using SL^2 projected onto the three principal components of the data (the red dots on the mesh).

The parameters of the methods under comparison were set as follows. WKNN used the same weighting scheme as in Eqn. (4) and performs a 4-nearest neighbor search in the training data. The nonlinear SVR made use of the RBF kernel. The parameters are selected with cross-validation (using the package libsvm [5]). The Biased Manifold method found a 100 dimensional embedding using 50 nearest neighbors (following [1]). The mapping from the input data to the embedded space was learned (as in [1]) using a Generalized Regression Neural Network regressor (GRNN). Finally, a weighted 4-nearest neighbor search is used in the embedded space to compute the angles for the test data. Our method (SL^2) uses 41 subspaces and 4 neighboring subspaces.

Figure 7 visualizes the means ($\mathbf{C}$) of the subspaces learned by SL^2 on the dataset **(I)** projected onto the three principal components of the data. The blue stars represent the training data, and the red dots the projects of the subspace means onto the first three principal components. It can be seen that SL^2 is very robust to the lack of uniform distribution in the training data. Quantitative results are summarized in the first two rows of Table 2. For the densely sampled dataset (Figure 6 (a)), our method outperforms existing approaches. Recall that WKNN and SVR required the storage of all 760 training data points, the biased manifold method needed to store 760 low-dimensional embeddings of the training data, while our method only uses 41 subspaces. For the sparsely sampled dataset (Figure 6 (b)) our method still gives the best MAE compared to the significantly degraded performance of all other methods. In both cases, our method still requires only 41 subspaces while other methods require all 79 training data points and produced worse errors. Note that the standard deviations for Dataset I are typically large, because some of the test images with angles close to $\pm 90^\circ$s are mistakenly associated to $\mp 90^\circ$s due to the high noise level in the data.

For dataset **(II)**, Figure 8 (first row), we selected the training set (blue crosses, 4188 images) containing only data points outside of the square region in which the 1812 (green dots) images were used for testing. In Figure 8 (second row), the space of angles, the red dots are the equidistantly resampled angles for training the subspaces in the input space. In this experiment, SL^2 used 300 subspaces (30 intervals along the yaw angles and 10 along the pitch angles) and 8 neighboring subspaces.

Figure 8 (third row) visualizes the means of the subspaces projected onto the first three principal components (the red dots on the mesh). The smoothness of the mesh indicates that the missing data, i.e., the green dots in Figure 8 (first row), has only a little effect on SL^2. The qualitative results are listed in the last row in Table 2. SL^2 produces the best MAE using only 300 subspaces and outperforms the other methods.

5. Conclusion and Future Work

This paper presents SL^2, a supervised generative method to learn a mixture of local subspaces that is robust to noise, lack of training samples and non-uniform sampled data. We have shown that SL^2 outperforms state-of-the-art methods, both in accuracy and computational complexity, on the problem of continuous head pose estimation using the BU-3DFE and CMU Multi-PIE database. One reason for the superior performance of SL^2 is that it performs a supervised resampling of the manifold which results in equidistantly spaced subspaces in both the input space and the output space. Throughout the paper, we have shown the application of SL^2 to the head pose estimation problem; however, SL^2 is a more general supervised learning algorithm and in future work we plan to apply it to other problems such as body pose estimation or facial expression synthesis.

Acknowledgements

The first authors was partially supported by National Science Foundation of China: "Vision Cognition and Intelligent Computation on Facial Expression" under Grant 60973070.

References

[1] V. Balasubramanian, Y. Jieping, and S. Panchanathan. Biased manifold embedding: A framework for person-independent head pose estimation. In *Proc. CVPR*, pages 1–7, 2007. 2922, 2926, 2927

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003. 2922

[3] D. Beymer. Face recognition under varying pose. In *Proc. CVPR*, pages 756–761, 1994. 2922

[4] V. Blanz, P. Grother, P. J. Phillips, and T. Vetter. Face recognition based on frontal views generated from non-frontal images. In *Proc. CVPR*, volume 2, pages 454– 461, 2005. 2921

[5] C. Chang and C. Lin. LIBSVM: a library for support vector machines. 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 2927

[6] T. Cootes, K. Walker, and C. Taylor. View-based active appearance models. In *Proc. IEEE Int'l. Conf. on Automatic Face and Gesture Recognition*, pages 227–232, 2000. 2922

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, volume 1, pages 886–893, 2005. 2923

[8] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, 2001. 2922

[9] Y. Fu and T. S. Huang. Graph embedded analysis for head pose estimation. In *Proc. IEEE Int'l. Conf. on Automatic Face and Gesture Recognition*, 2006. 2922, 2926

[10] A. Gee and R. Cipolla. Determining the gaze of faces in images. *Image and Vision Computing*, 12(10):639–647, 1994. 2922

[11] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-PIE. *Image and Vision Computing*, 28(5):807–813, 2010. 2922, 2925

[12] J. Huang, X. Shao, and H. Wechsler. Face pose discrimination using support vector machines. In *Proc. ICPR*, pages 154–156, 1998. 2922

[13] M. Jones and P. Viola. Fast multi-view face detection. *Technical Report 096, Mitsubishi Electric Research Laboratories*, 2003. 2922

[14] S. R. H. Langton, H. Honeyman, and E. Tessler. The influence of head contour and nose angle on the perception of eye-gaze direction. *Perception and Psychophysics*, 66(5):752–771, 2004. 2921

[15] S. Li, Q. Fu, L. Gu, B. Scholkopf, Y. Cheng, and H. Zhang. Kernel machine based learning for multi-view face detection and pose estimation. In *Proc. ICCV*, pages 674–679, 2001. 2922

[16] Y. Li, S. Gong, J. Sherrah, and H. Liddell. Support vector machine based multi-view face detection and recognition. *Image and Vision Computing*, 22(5):413–427, 2004. 2922

[17] G. Little, S. Krishna, J. Black, , and S. Panchanathan. A methodology for evaluating robustness of face recognition algorithms with respect to variations in pose and illumination angle. In *Proc. IEEE Int'l. Conf. on Acoustics, Speech and Signal Processing*, pages 89–92, 2005. 2923

[18] Y. Ma, Y. Konishi, K. Kinoshita, S. Lao, and M. Kawade. Sparse bayesian regression for head pose estimation. In *Proc. ICPR*, pages 507–510, 2006. 2922, 2926

[19] S. J. McKenna and S. Gong. Real-time face pose estimation. *Real-Time Imaging*, 4(5):333–347, 1998. 2922

[20] H. Moon and M. Miller. Estimating facial pose from a sparse representation. In *Proc. ICIP*, pages 75–78, 2004. 2922

[21] E. Murphy-Chutorian and M. M. Trivedi. HyHOPE: hybrid head orientation and position estimation for vision-based driver head tracking. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 512–517, 2008. 2921

[22] E. Murphy-Chutorian and M. M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Trans. on PAMI*, 31(4):607–626, 2009. 2922

[23] S. Niyogi and W. Freeman. Example-based head tracking. In *Proc. IEEE Int'l. Conf. on Automatic Face and Gesture Recognition*, pages 374–378, 1996. 2922

[24] A. Ranganathan and M.-H. Yang. Online sparse matrix gaussian process regression and vision applications. In *Proc. ECCV*, 2008. 2922, 2926

[25] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 2922

[26] J. Tenenbaum, V. Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 2922

[27] J. Wu and M. Trivedi. A two-stage head pose estimation framework and evaluation. *Pattern Recognition*, 41(3):1138–1158, 2008. 2922

[28] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato. A 3D facial expression database for facial behavior research. In *Proc. IEEE Int'l. Conf. on Automatic Face and Gesture Recognition*, pages 211 – 216, 2006. 2922, 2925, 2926