Fast Transformation-Invariant Component Analysis

Anitha Kannan[1], **Nebojsa Jojic**[2], **Brendan J. Frey**[1]

[1] University of Toronto, http://www.psi.toronto.edu

[2] Microsoft Research, http://research.microsoft.com/∼jojic

For software and more illustrations: http://www.psi.utoronto.ca/∼anitha/fastTCA.htm

Abstract

Dimensionality reduction techniques such as principal component analysis and factor analysis are used to discover a linear mapping between high dimensional data samples and points in a lower dimensional subspace. Previously, transformation-invariant component analysis (TCA) was introduced to learn this linear mapping in a way that is invariant to a set of global transformations. The expectation maximization algorithm used to learn the parameters of TCA requires a number of scalar operations on the order of N^2, where N is the number of elements in each training example. This is prohibitive for many applications of interest such as modelling mid- to large- size images, where N may be quite large (e.g. 262144 dimensions for a 512×512 grayscale image). In this paper, we present an efficient algorithm that reduces the computational requirements to the order of $N \log N$. With this speedup, we show the effectiveness of TCA in various applications including tracking, video textures, clustering, object recognition and object detection in images.

I. Introduction

Methods such as principal component analysis [13] and factor analysis [4] are widely used to linearly project high dimensional data samples onto points in a lower dimensional subspace. In computer vision, these models are used in applications such as modelling faces to learn facial expressions (e.g. [8] , [18]), learning representation of images of handwritten digits [12] and modelling appearance changes in reasonably long video sequences (e.g. [1], [2]). An underlying assumption of these linear dimensionality reduction methods is that the amount of interesting variation is small or local to be faithfully explained in a linear subspace.

A linearized manifold, however, is insufficient to capture large variations such as global transformations (e.g. shifts, rotations and scales) present in the data. As an example, in fig. 1a, we see that the basis vectors of a factor analyzer trained on a walk sequence models a rather poor linearized manifold of the global shifts as opposed to more interesting periodic leg movements. Often, a preprocessing step is employed to align data points prior to performing linear dimensionality reduction (c.f. [18] in

the context of learning eigen faces) to a reference, which is usually a data point from the training set. In fig. 1b we show the basis vectors of the factor analysis model learned on the same data set used in fig. 1a but pre-processed so that the images are aligned. Even though the learned basis vectors look much better than those learned without preprocessing, the reconstruction of the training set from the model using model parameters and the inferred latent variables is noisy (e.gs. in fig. 3b). This is because pre-alignment transforms the location of noise as well. In addition, in the presence of large amount of noise such as background clutter none of the data points can serve as a good reference. Hence, a better approach is to jointly estimate global transformations and learn a data representation invariant to transformations. In [5], it is shown that inferring transformation while clustering of 2D images provided better 2D clusters. Similarly, in [3] it is shown that time-varying multidimensional curves such as expression profile of genes measured over time, and measurement of tropical cyclone trajectories are better clustered when transformations are modelled explicitly and jointly within the clustering framework.

We can model transformation directly on the n dimensional feature space representing the data. This approach is robust to sensor noise and other kinds of noise such as background clutter. However, even a single directional shift of the feature vector traces a 1D curve in the transformation manifold, and in general when a feature vector is acted upon by a set of transformations with k parameters, it traces a k dimensional curve in the n dimensional feature space (c.f. [9]). Therefore, finding the exact optimal location on this non-linear manifold that provides a perfect alignment of data points is infeasible, and instead an approximation is typically used.

In [5], a discrete approximation to the transformation manifold is proposed for transformation invariant probabilistic clustering of data points. In an extension to this work [6], data points are more succinctly explained in a linear subspace while accounting for global transformations. The resulting model, called transformation invariant component analysis (TCA) can learn subspace representations invariant to a set of predefined global transformations. In general, TCA can be used as a module in models that require learning linear subspaces while simultaneously accounting for transformations. For example, when modelling a video sequence as multiple layers of moving objects, TCA can be used to model objects in each layer. This enables capturing the appearance variability in a low dimensional subspace while accounting for large translation in the location of the object [10].

A significant limitation with the previously published technqiues for performing TCA is that it is computationally expensive for performing inference and learning. In fact, the EM algorithm outlined in [6] has a computational complexity of $K^2|\mathcal{T}|N$ where K is the number of factors (or linear basis), N is the number of elements in each training case and $\mathcal{T}$ is the set of all possible transformations. This means that it is restricted to be used only on low resolution data. If we consider all possible shifts, then $|\mathcal{T}| = N$ and the complexity becomes K^2N^2.

In this work, we present an efficient inference and learning algorithm that has complexity of only $K^2N \log N$ [14]. This

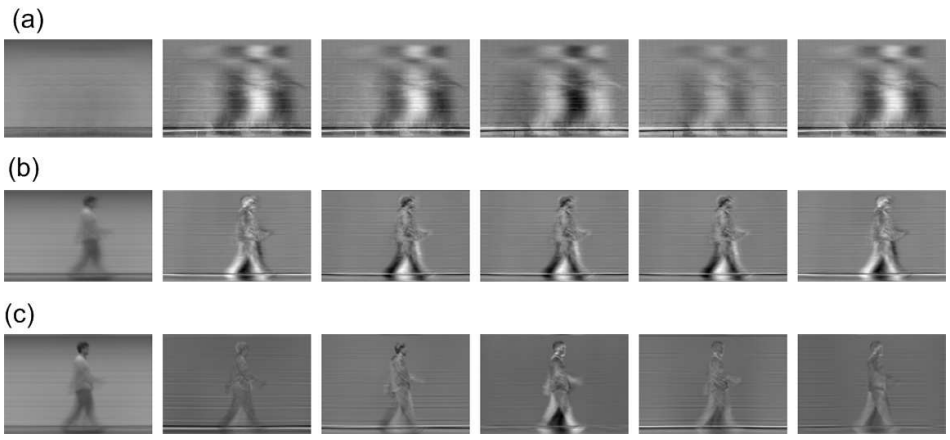

Fig. 1. Means μ and components in $\mathbf{\Lambda}$ learned using (a) FA, (b) FA applied on data normalized using a correlation tracker, and (c) transformed component analysis (TCA) applied directly on data.

approach does not sacrifice the exactness of the inference - our proposed algorithm produces the same posterior distribution obtained using brute-force evaluation. The main computational overhead in TCA involves matching a signal with all possible shifts of another signal. This is a basic problem in signal processing and can be efficiently solved by applying the fast Fourier transform (FFT). This observation has been used before for learning transformation invariant mixture of Gaussians model [7]. Here, we use this and mild assumptions on the original model to present an effective implementation. For instance, our implementation will be 21000 times faster than the original TCA algorithm on a 512×512 image. We believe that this derivation of EM algorithm for performing TCA will broaden the application areas to which it can be applied. To this end, we present experimental results in various computer vision applications including tracking, video textures, clustering video sequences, object recognition and object detection. Note that the data sets used in these experiments can not be processed using TCA model without the tremendous speedup we have obtained using this proposed work.

II. Transformation-invariant component analysis and its mixture model

Transformation-invariant component analysis (TCA) [6] is a generative model based method for performing transformation-invariant linear dimensionality reduction. The set of transformations, $\mathcal{T}$, to which the model is invariant to is specified a priori. Fig. 2a. shows the generative model for TCA. The K dimensional subspace vector $\mathbf{y}$ is drawn from the standard Gaussian distribution. $\mathbf{y}$ is projected to N dimensional latent image, $\mathbf{z}$ using the $N \times K$ factor loading matrix $\mathbf{\Lambda}$ and shifting the origin by μ. This projection is noisy, and we assume that the noise is Gaussian with zero mean, and diagonal covariance, $\mathbf{\Psi}$. The columns of $\mathbf{\Lambda}$ represent the basis vectors that span the linear subspace. A N dimensional observation $\mathbf{x}$ is obtained by applying a transformation $\mathbf{T} \in \mathcal{T}$, distributed according to $\rho_{\mathbf{T}}$ on the latent image $\mathbf{z}$ and adding independent Gaussian noise

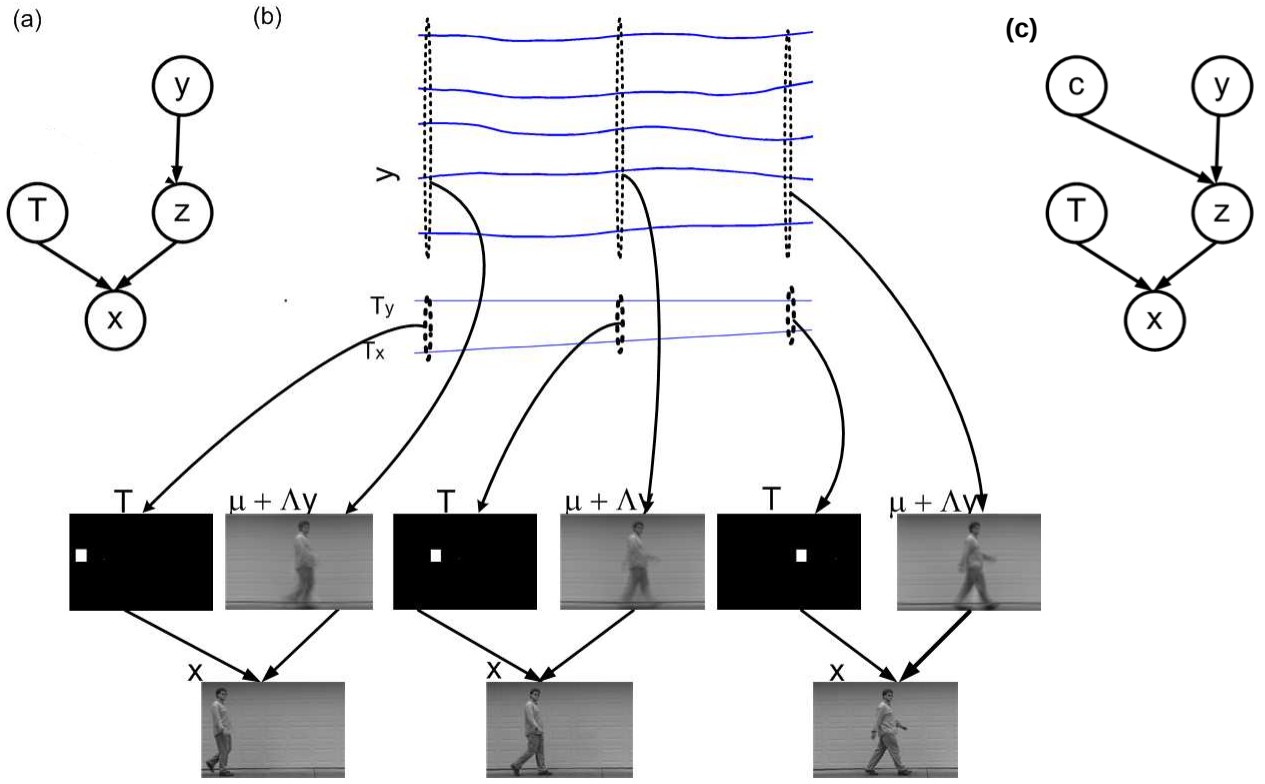

Fig. 2. Mixture of transformed component analyzers (MTCA). (a) The generative model with cluster index c, subspace coordinates $\mathbf{y}$, latent image $\mathbf{z} = \mu_{\mathbf{c}} + \mathbf{\Lambda_c y}$+noise; transformation $\mathbf{T}$ and generated final image $\mathbf{x} = \mathbf{Tz}$+noise; (b) An example of the generative process, where subspace coordinates $\mathbf{y}$, and image position $\mathbf{T_x}$, $\mathbf{T_y}$ are inferred from a captured video sequence $\mathbf{x}$

with covariance $\mathbf{\Psi}$. The joint distribution over all variables is ([6]):

$$p(\mathbf{x}, \mathbf{T}, \mathbf{z}, \mathbf{y}) = p(\mathbf{x}|\mathbf{z}, \mathbf{T})\mathbf{p}(\mathbf{z}|\mathbf{y})\mathbf{p}(\mathbf{y})\mathbf{P}(\mathbf{T}) = \mathcal{N}(\mathbf{x}; \mathbf{Tz}, \mathbf{\Psi})\mathcal{N}(\mathbf{z}; \mu + \mathbf{\Lambda y}, \mathbf{\Phi})\mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I})\rho_{\mathbf{T}} \quad (1)$$

Here, the operation $\mathbf{Tz}$ defines the transformed vector obtained by applying the transformation $\mathbf{T}$ on $\mathbf{z}$. We interchangeably use $\mathbf{T}$ to represent chosen discrete transformation and also the transformation matrix (corresponding to transformation $\mathbf{T}$) that transforms the vector acting right of it. Fig. 2b illustrates the generative process when the set of transformations is discrete shifts. Here, the subspace coordinates $\mathbf{y}$ are used to generate a latent image $\mathbf{z}$ (without noise) and the horizontal and vertical image position $\mathbf{T_x}$ and $\mathbf{T_y}$ are used to shift the latent image to obtain $\mathbf{x}$. In fact, $\mathbf{y}$, $\mathbf{T_x}$ and $\mathbf{T_y}$ shown in the figure are actually inferred from the captured video sequence $\mathbf{x}$ (see sec. V-A).

The generative framework provides an elegant way to incorporate the knowledge that each observation can be generated from one of many different classes of objects. This is important when we want to cluster the data and learn linear manifold for each cluster invariant to transformations. This is useful, for instance, in learning appearance models of facial expressions of different people using an unlabelled data set in which the location of the face in each image varies - here, the clusters represent different people, the subspace for each cluster models local variations in facial expressions, and the global transformations

handle large shift in the location of the face in each image. For this, we incorporate a class variable into the generative model for TCA as shown in Fig. 2c. This resulting mixture model is known as a mixture of transformation-invariant component analysis (MTCA), and it enables performing transformation-invariant clustering and learning a subspace representation for each cluster. The generative process is similar to TCA, except that we sample the class from the prior discrete probability π_c and use the basis vectors and the mean for that sampled class to generate the latent image $\mathbf{z} = \mu_{\mathbf{c}} + \mathbf{\Lambda_c y} + \text{noise}$. In this case, the joint distribution is:

$$p(\mathbf{x}, \mathbf{T}, \mathbf{z}, \mathbf{c}, \mathbf{y}) = p(\mathbf{x}|\mathbf{z}, \mathbf{T})\mathbf{p}(\mathbf{z}|\mathbf{y}, \mathbf{c})\mathbf{p}(\mathbf{y})\mathbf{P}(\mathbf{T})\mathbf{P}(\mathbf{c}) = \mathcal{N}(\mathbf{x}; \mathbf{T}\mathbf{z}, \mathbf{\Psi})\mathcal{N}(\mathbf{z}; \mu_{\mathbf{c}} + \mathbf{\Lambda_c y}, \mathbf{\Phi_c})\mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I})\rho_{\mathbf{T}}\pi_{\mathbf{c}}. \tag{2}$$

Note that when the number of classes is one, this model reduces to the TCA model. Hence, we will focus on the more general MTCA model. Since all distributions in the model are either Gaussians or discrete and interactions between variables are linear, we can marginalize over variables that are not of interest. For instance, the marginal distribution $p(\mathbf{x}, \mathbf{c}, \mathbf{T})$ involves integrating over $\mathbf{z}$ and $\mathbf{y}$ so that:

$$p(\mathbf{x}, \mathbf{T}, \mathbf{c}) \quad = \quad \int_{\mathbf{z}} \int_{\mathbf{y}} p(\mathbf{x}, \mathbf{T}, \mathbf{z}, \mathbf{c}, \mathbf{y}) = \mathcal{N}(\mathbf{x}; \mathbf{T}\mu_{\mathbf{c}}, \mathbf{T}(\mathbf{\Lambda_c \Lambda_c'} + \mathbf{\Phi_c})\mathbf{T}' + \mathbf{\Psi})\rho_{\mathbf{T}}\pi_{\mathbf{c}} \tag{3}$$

Thus, for every transformation $\mathbf{T}$ and class c there is a corresponding mean image $\mu_{\mathbf{c}}$ and covariance matrix, $\mathbf{T}(\mathbf{\Lambda_c \Lambda_c'} + \mathbf{\Phi_c})\mathbf{T}' + \mathbf{\Psi}$. The covariance, $\mathbf{\Lambda_c \Lambda_c'} + \mathbf{\Phi_c}$, of the latent image (obtained by integrating out the subspace explanation) and the observation noise $\mathbf{\Psi}$ add after the covariance of latent image is transformed to the frame of the observed image. The probability of observation is given by

$$p(\mathbf{x}) = \sum_{\mathbf{c=1}}^{\mathbf{C}} \sum_{\mathbf{T} \in \mathcal{T}} \mathcal{N}(\mathbf{x}; \mathbf{T}\mu_{\mathbf{c}}, \mathbf{T}(\mathbf{\Lambda_c \Lambda_c'} + \mathbf{\Phi_c})\mathbf{T}' + \mathbf{\Psi})\rho_{\mathbf{T}}\pi_{\mathbf{c}} \tag{4}$$

The parameters of the model $\{\mu_{\mathbf{c}}, \mathbf{\Phi_c}, \mathbf{\Lambda_c}, \mathbf{\Psi}\}_{\mathbf{c=1}}^{\mathbf{C}}$ are learned from a set of J i.i.d training examples $\{\mathbf{x}^{(\mathbf{j})}\}_{\mathbf{j=1}}^{\mathbf{J}}$ by maximizing the parameters likelihood ($\prod_{j=1}^{j=J} p(\mathbf{x}^{(\mathbf{j})})$) using an exact EM algorithm. The only inputs to the EM are the training examples, the number of factors, K, the number of clusters, C, and the set of all possible transformations, $\mathcal{T}$. Starting at a random initialization, EM algorithm for MTCA iterates between E step, where it probabilistically fills in for hidden variables by finding the exact posterior $p(\mathbf{y}, \mathbf{z}, \mathbf{c}, \mathbf{T}|\mathbf{x})$ and M step where it updates the parameters.

The probability of data (eqn. 4) requires summing over all possible transformations and classes and hence is very expensive. In fact, each of the inference and update equations in [6] has a complexity of $K^2|\mathcal{T}|N$. When we consider all possible shifts of the signal, this complexity is equivalent to K^2N^2. In the next section, we present a fast exact EM algorithm for performing inference and learning using very mild assumptions. The proposed algorithm is derived and evaluated in Fourier domain at a considerably lower computational cost of $K^2N \log N$. As the number of factors K is usually very small, the complexity is in the order of only $N \log N$.

III. Assumptions and notation for fast exact EM

In this section, we discuss basic assumptions that enable fast inference and learning in the TCA model. We also describe the notation used in the rest of the paper.

A. Assumptions

We assume that data is represented in a coordinate system in which transformations are discrete shifts with wrap-around. For translations, this corresponds to a 2D rectangular grid coordinate system. Similarly, rotations and scales correspond to shifts in a radial grid (c.f.[7] [19]). For notational convenience, we assume that the transformation is orthogonal and invertible such that $\mathbf{T}'\mathbf{T} = \mathbf{T}\mathbf{T}' = \mathbf{I}$.

We also assume that the post-transformation noise is isotropic, $\mathbf{\Psi} = \psi\mathbf{I}$, so that covariances matrices such as $\mathtt{COV}[\mathbf{z}|\mathbf{T}, \mathbf{c}, \mathbf{x}]$ and $\mathtt{COV}[\mathbf{y}|\mathbf{T}, \mathbf{c}, \mathbf{x}]$ are independent of $\mathbf{T}$, simplifying computations. This assumption introduces ambiguity in differentiating between pre- and post- transformation noise ($\mathbf{\Phi}$ and $\mathbf{\Psi}$), but our experimental evaluation suggests that in many applications performance is not hindered. In fact, for isotropic $\mathbf{\Psi}$, it is possible to preset ψ to a small value so that if the actual value in the data is larger it can be accounted for in $\mathbf{\Phi}$. In our experiments we set ψ to .001.

B. Notation

We describe the notation that simplifies expressions for transformation that corresponds to a shift in input coordinates. Let $\mathbf{i}$ be an integer vector in the coordinate system in which input is measured. For 2D $n \times m$ image, ($N = n \times m$), $\mathbf{x}(\mathbf{i})$ is the $\mathbf{i}^{th}$ element where $\mathbf{i} \in \{(i_1, i_2) : i_1 = 1 \cdots n, i_2 = 1 \cdots m\}$. Vectors in the input coordinate system such as $\mathbf{x}, \mu, \mathbf{z}$ are defined this way. For diagonal matrices such as $\mathbf{\Phi}$, $\mathbf{\Phi}(\mathbf{i})$ defines the element corresponding to the pixel at coordinate $\mathbf{i}$. This enables treating transformations corresponding to a shift be represented as a vector $\mathbf{T}$ in the same system, so that a shift of $\mathbf{z}$ by $\mathbf{T}$ is represented as $\mathbf{z}(\mathbf{i} + \mathbf{T})$ such that $\mathbf{i} + \mathbf{T}$ = ($i_1 + T_1$ mod n, $i_2 + T_2$ mod n).

We show that *all* expensive operations for inference and learning involve correlation, or convolution. When computed in the pixel domain, these operations take $O(|\mathcal{T}|N)$. However, convolution and correlation can be computed in the frequency domain with computational cost of $O(|\mathcal{T}| \log N)$, i.e. $O(N \log N)$ for all shifts:

$$Correlation : [\mathbf{p} \odot \mathbf{q}] = \sum_{\mathbf{i}=1}^{N} \mathbf{p}(\mathbf{i})\mathbf{q}(\mathbf{i} - \mathbf{T}) = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{p})\overline{\mathcal{F}(\mathbf{q})}) \tag{5}$$

$$Convolution : [\mathbf{p} \otimes \mathbf{q}] = \sum_{\mathbf{T}\in\mathcal{T}} \mathbf{p}(\mathbf{T})\mathbf{q}(\mathbf{i} - \mathbf{T}) = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{p})\mathcal{F}(\mathbf{q})) \tag{6}$$

where, $\mathcal{F}$ and $\mathcal{F}^{-1}$ are two-dimensional Fourier and inverse Fourier transforms and $\overline{\mathcal{F}(\mathbf{q})}$ is the complex conjugate of $\mathcal{F}(\mathbf{q})$. We represent k^{th} column of a matrix, $\mathbf{V}$, by $(\mathbf{V})_{(:,k)}$ and the k^{th} row by $(\mathbf{V})_{(k,:)}$. Also, diag($\mathbf{V}$) extracts the diagonal elements of matrix $\mathbf{V}$ and the operation $\mathbf{p} \circ \mathbf{q}$ defines the element wise product between vectors $\mathbf{p}$ and $\mathbf{q}$.

IV. Fast inference and learning in MTCA using EM algorithm

Here, we present a fast Expectation Maximization algorithm for learning the mixture of transformed component analyzers. The only input to the algorithm is a set of J observations (training examples), $\{\mathbf{x^{(j)}}\}_{\mathbf{j=1}}^{\mathbf{J}}$, the set of possible transformations, $\mathcal{T}$, the number of clusters C and the dimensionality of the subspace K. The algorithm starts with random initialization of the parameters, $\{\mu_{\mathbf{c}}, \mathbf{\Lambda_c}, \mathbf{\Phi_c}\}_{\mathbf{c=1}}^{\mathbf{C}}$, and iterates between finding the distribution over the unobserved variables in the model, $p(c, \mathbf{T}, \mathbf{z}, \mathbf{y}|\mathbf{x^{(j)}})$ for each training example, j and finding the maximum likelihood parameters. MATLAB software is available at www.psi.utoronto.ca

Initialization of parameters: Expectation maximization algorithm is prone to local optimal solutions. To obtain better optimal solutions and faster convergence, we initialize the parameters as follows: The means $\{\mu_{\mathbf{c}}\}_{\mathbf{c=1}}^{\mathbf{C}}$ are initialized to the mean of the training set. To break symmetry between different class means, zero mean Gaussian noise with variance equal to variance of the data is added to the means. The diagonal covariance matrices $\{\mathbf{\Phi_c}\}_{\mathbf{c=1}}^{\mathbf{C}}$ of the latent space are set to two times the variance of the data. The variance of the observation ψ is fixed at .001. The values of factor loading matrices, $\{\mathbf{\Lambda_c}\}_{\mathbf{c=1}}^{\mathbf{C}}$ is set to random numbers generated from a Gaussian with mean 0 and variance .01. This initialization of factor loading matrices ensures that the values are not biased towards positive values and are close to 0 so as to escape initial bad local optimal subspace representation due to broad posterior distributions. In fact, in our experiments, we first perform 5 iterations of transformed mixture of Gaussians by clamping the values of $\{\mathbf{\Lambda_c}\}_{\mathbf{c=1}}^{\mathbf{C}}$ to 0, and then initialize the factor loading matrices as described.

The EM algorithm proceeds by iterating between the Estep where the posterior distribution over the unobserved variables $p(\mathbf{z}, \mathbf{y}, \mathbf{T}, \mathbf{c}|\mathbf{x})$ (sec. IV-B) is computed for the current setting of the model parameters and the Mstep where the maximum likelihood parameters (sec. IV-A) is found by maximizing the expected complete log likelihood under the posterior distribution obtained in the Estep. The convergence of the EM is decided by monitoring the increase in the (log)likelihood function.

A. Maximization step

In the Mstep, the model parameters are updated so as to maximize the expected complete log-likelihood of the parameters, where the expectation is under the posterior distribution of the unobserved variables. This optimization involves updating each parameter to a value that satisfies:

$$\sum_{j=1}^{J}\sum_{c=1}^{C}\sum_{\mathbf{T}\in\mathcal{T}}\int_y\int_z |d\mathbf{y}||\mathbf{dz}|\mathbf{p}(\mathbf{c}, \mathbf{T}, \mathbf{z}, \mathbf{y}|\mathbf{x^{(j)}})\frac{\partial}{\partial\theta}\log\mathbf{p}(\mathbf{x^{(j)}}, \mathbf{z}, \mathbf{y}, \mathbf{T}, \mathbf{c}) = \mathbf{0} \tag{7}$$

Here θ is one of $\pi_c, \mu_{\mathbf{c}}, \mathbf{\Phi_c}, \mathbf{\Psi}$ and the joint distribution $p(\mathbf{x^{(j)}}, \mathbf{z}, \mathbf{y}, \mathbf{T}, \mathbf{c})$ is given by eqn. 1. Thus, we take the derivative with respect to a parameter, numerically average it over the hidden variables using the posterior distribution $p(c, \mathbf{T}, \mathbf{z}, \mathbf{y}|\mathbf{x^{(j)}})$ for each training example. In this section we describe the parameter updates.

For each class index, the prior probability of that class π_c is given by the average of the posterior distribution $P(c|\mathbf{x^{(j)}})$ (computed as described in sec. IV-B.5) over the class index for each training example:

$$\pi_c \leftarrow \frac{1}{J}\sum_{j=1}^{J} P(c|\mathbf{x^{(j)}})$$

The prior distribution over the transformations is assumed to be uniform as often there is insufficient amount of data to learn its parameters. For instance, for a N dimensional image, there are N possible shift transformations and in this case learning a prior distribution without overfitting will typically require a data set with larger than N examples.

The mean of the latent image for each cluster is given by the average value of the weighted difference in the posterior mean of the latent image and the projection onto the subspace corresponding to that class. This weight is dictated by the posterior probability of that class given the observation.

$$\mu_{\mathbf{c}} \leftarrow \frac{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})E[z - \Lambda_c y|c, x^{(j)}]}}{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})}} = \frac{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})}\Big(\mathbf{E[z|c, x^{(j)}] - \Lambda_c E[y|c, x^{(j)}]}\Big)}{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})}}$$

The posterior mean of latent image, $E[\mathbf{z}|\mathbf{c}, \mathbf{x^{(j)}}]$ and the subspace projection, $E[\mathbf{y}|\mathbf{c}, \mathbf{x^{(j)}}]$ for each class and observation can be computed as described in sec. IV-B.1 & IV-B.2. The value of $\mathbf{\Lambda_c}$ from the previous iteration or its most recent value can be used, and both approach will increase the probability of observations.

The pre-transformation noise is given by the weighted average of mean squared error between the inferred latent image and the expected model prediction (IV-B.4):

$$\mathbf{\Phi_c} \leftarrow \frac{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})E}\Big[\mathbf{diag}\big((\mathbf{z} - \mu_c - \mathbf{\Lambda_c y})(\mathbf{z} - \mu_c - \mathbf{\Lambda_c y})'\big)|\mathbf{x^{(j)}}, \mathbf{c}\Big]}{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})}}$$

Again, value of $\mathbf{\Lambda_c}$ and $\mu_{\mathbf{c}}$ from the previous iteration or the most recently updated value can be used.

The factor loading matrix for each class is updated as:

$$\mathbf{\Lambda_c} \leftarrow \left(\frac{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})E[zy'|c, x^{(j)}]} - \mu_{\mathbf{c}}\mathbf{E[y|c, x^{(j)}]}}{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})}}\right)\left(\frac{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})E[yy'|c, x^{(j)}]}}{\sum_{\mathbf{j=1}}^{\mathbf{J}} \mathbf{P(c|x^{(j)})}}\right)^{-1}$$

At this juncture, we would like to point out that the update equations for the model parameters are exactly same as in the case of a regular mixture of factor analysis model ([11]), except that the expected values required in this case are obtained after normalizing for transformations. Therefore, the computations involved in the Mstep are on par with the computational requirements of the regular mixture of factor analysis model.

In the next section, we describe the computation involved in performing inference. Here, we leverage on our assumptions and the use of FFTs to obtain a tremendous decrease in computational requirements.

B. Inference

Inference refers to finding the posterior distribution of the unobserved variables given instantiations of some of the variables. In our model, the posterior distribution is $p(\mathbf{z}, \mathbf{y}, \mathbf{T}, \mathbf{c}|\mathbf{x})$ and using the chain law of probabilities, this can be represented as:

$$p(\mathbf{z}, \mathbf{y}, \mathbf{T}, \mathbf{c}|\mathbf{x}) = \mathbf{p}(\mathbf{z}|\mathbf{y}, \mathbf{x}, \mathbf{T}, \mathbf{c})\mathbf{p}(\mathbf{y}|\mathbf{x}, \mathbf{T}, \mathbf{c})\mathbf{P}(\mathbf{T}, \mathbf{c}|\mathbf{x}) \tag{8}$$

As interactions of all variables in our generative model are linear, and the distributions are Gaussian, the posterior distributions, $p(\mathbf{z}|\mathbf{y}, \mathbf{x}, \mathbf{T}, \mathbf{c})$ and $p(\mathbf{y}|\mathbf{x}, \mathbf{T}, \mathbf{c})$ are Gaussians. As transformation $\mathbf{T}$ and cluster variable c are discrete, $P(\mathbf{T}, \mathbf{c}|\mathbf{x})$ is a discrete distribution.

1) Inferring the latent image: Given $\mathbf{x}, \mathbf{T}$ and c, the mean of the latent image is the weighted combination of the model prediction and the observation:

$$E[\mathbf{z}|\mathbf{x}, \mathbf{T}, \mathbf{c}] = \mathtt{COV}[\mathbf{z}|\mathbf{x}, \mathbf{T}, \mathbf{c}]\Big((\mathbf{\Lambda_c}\mathbf{\Lambda_c}' + \mathbf{\Phi_c})^{-1}\mu_\mathbf{c} + \mathbf{\Psi}^{-1}\mathbf{T}'\mathbf{x}\Big) \tag{9}$$

The model prediction term $(\mathbf{\Lambda_c}\mathbf{\Lambda_c}' + \mathbf{\Phi_c})^{-1}\mu_\mathbf{c}$ weights the cluster mean by the inverse covariance in the latent space, and the observation term $\mathbf{\Psi}^{-1}\mathbf{T}'\mathbf{x}$ scales the transformation normalized observation $\mathbf{T}'\mathbf{x}$ according to the inverse of the observation noise. Therefore, the contribution of the model term and the observation term is determined by the level of pre and post transformation noise respectively. The covariance $\mathtt{COV}[\mathbf{z}|\mathbf{x}, \mathbf{T}, \mathbf{c}]$ matrix given by:

$$\mathtt{COV}[\mathbf{z}|\mathbf{x}, \mathbf{T}, \mathbf{c}] = \Big((\mathbf{\Lambda_c}\mathbf{\Lambda_c}' + \mathbf{\Phi_c})^{-1} + \mathbf{\Psi}^{-1}\Big)^{-1} = \mathbf{D} + \mathbf{EF}$$

is the inverse of sum of inverse covariance of latent space (includes the covariance of projecting from the subspace) and the inverse covariance of the observation. As can be seen, it is independent of $\mathbf{T}$ which is due to the assumption of isotropic post-transformation noise and the property of the transformation operation $\mathbf{T}\mathbf{T}' = \mathbf{T}'\mathbf{T} = \mathbf{I}$ [1]. The factorization of the matrix into $\mathbf{D} + \mathbf{EF}$ is obtained through multiple applications of matrix inversion lemma and it enables computing covariance matrix without any non-diagonal matrix inversion. Here, $\mathbf{E} = \mathbf{D}\mathbf{\Phi_c}^{-1}\mathbf{\Lambda_c}\mathbf{B}$ and $\mathbf{F} = \mathbf{\Lambda_c}'\mathbf{\Phi_c}^{-1}\mathbf{D}$ are $N \times K$ and $K \times N$ matrices and $\mathbf{D} = \mathtt{COV}[\mathbf{z}|\mathbf{x}, \mathbf{y}, \mathbf{T}, \mathbf{c}] = (\mathbf{\Phi_c}^{-1} + \mathbf{\Psi}^{-1})^{-1}$ is a diagonal matrix. At this juncture, we point out that the covariance matrix need not be explicitly computed during learning (in fact no $N \times N$ matrix is ever computed) as it is often right multiplied with a vector (e.g. eqn. 9). Therefore, we can make use of the factorization and perform right to left multiplication.

The posterior over the latent image, $p(\mathbf{z}|\mathbf{x}, \mathbf{c}) = \sum_{\mathbf{T}\in\mathcal{T}} \mathbf{P}(\mathbf{T}|\mathbf{x}, \mathbf{c})\mathbf{p}(\mathbf{z}|\mathbf{x}, \mathbf{T}, \mathbf{c})$ after marginalizing over $\mathbf{T}$ is a mixture of $\mathcal{T}$ Gaussians with mixing proportions, $P(\mathbf{T}|\mathbf{x}, \mathbf{c})$ and hence the posterior distribution can be multimodal when significantly different transformations are appropriate. During initial stages of learning, the posterior over $\mathbf{T}$ is typically a broad distribution.

[1]Without these assumptions, $\mathtt{COV}[\mathbf{z}|\mathbf{x}, \mathbf{T}, \mathbf{c}] = ((\mathbf{\Lambda_c}\mathbf{\Lambda_c}' + \mathbf{\Phi_c})^{-1} + \mathbf{T}\mathbf{\Psi}^{-1}\mathbf{T}')^{-1}$ as described in [6], in which case this covariance matrix needs to be evaluated for all possible transformations

As finding the mode of a mixture distribution is quite difficult, we can use its average, which is the mean of the posterior:

$$\begin{aligned} E[\mathbf{z}|\mathbf{x},\mathbf{c}] &= \sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}|\mathbf{x},\mathbf{c})\mathbf{E}[\mathbf{z}|\mathbf{x},\mathbf{T},\mathbf{c}] \\ &= \mathtt{COV}[\mathbf{z}|\mathbf{x},\mathbf{c}](\mathbf{\Lambda_c}\mathbf{\Lambda_c^T}+\mathbf{\Phi_c})^{-1}\mu_\mathbf{c} + \mathtt{COV}[\mathbf{z}|\mathbf{x},\mathbf{c}]\mathbf{\Psi^{-1}}\sum_{\mathbf{T}\in\mathcal{T}}\mathbf{P}(\mathbf{T}|\mathbf{c},\mathbf{x})\mathbf{T}^{'}\mathbf{x} \end{aligned}$$

The first term of this sum is dictated by the model and can be easily computed by first factorizing the matrix inverse, using the factorization of the covariance matrix and applying right to left multiplication. The sum,$\sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}|\mathbf{c},\mathbf{x})\mathbf{T}^{'}\mathbf{x}$ is a convolution of $\mathbf{x}$ with the probability map $P(\mathbf{T}|\mathbf{c},\mathbf{x})$ defined for all $\mathbf{T}$. This is clear by looking at a particular element in the sum which is given by $\sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}|\mathbf{c},\mathbf{x})\mathbf{x}(\mathbf{i}-\mathbf{T})$. Using eqn. 6, we can efficiently compute this sum for all $\mathbf{i}$ in the frequency domain so that

$$E[\mathbf{z}|\mathbf{c},\mathbf{x}] = \mathtt{COV}[\mathbf{z}|\mathbf{x},\mathbf{c}](\mathbf{\Lambda_c}\mathbf{\Lambda_c^{'}}+\mathbf{\Phi_c})^{-1}\mu_\mathbf{c} + \mathtt{COV}[\mathbf{z}|\mathbf{x},\mathbf{c}]\mathbf{\Psi^{-1}}\Big[\mathbf{P}(\mathbf{T}|\mathbf{c},\mathbf{x})\otimes\mathbf{x}\Big]$$

2) Inferring the subspace coordinates: In PCA where there is no model for noise, the data is projected to subspace through the projection matrix. In the same spirit, using our assumption that $\mathbf{T}\mathbf{T}^{'}=\mathbf{I}$ and $\mathbf{\Psi}=\psi\mathbf{I}$, we can derive that the subspace projection matrix for each class of MTCA model is $\mathbf{Q_c}=\mathtt{COV}[\mathbf{y}|\mathbf{x},\mathbf{T},\mathbf{c}]\mathbf{\Lambda_c'}(\mathbf{\Phi_c}+\mathbf{\Psi})^{-1}$. This projection matrix accounts for the inverse of noise variances in projected subspace through $\mathtt{COV}[\mathbf{y}|\mathbf{x},\mathbf{T},\mathbf{c}]=\Big(\mathbf{I}+\mathbf{\Lambda_c'}(\mathbf{\Phi_c}+\mathbf{\Psi})^{-1}\mathbf{\Lambda_c'}\Big)^{-1}$ and the inverse of noise variance in input space through $(\mathbf{\Phi_c}+\mathbf{\Psi})^{-1}$, which is projected to subspace by pre-multiplying with $\mathbf{\Lambda_c'}$. Thus we project transformation normalized $\mathbf{x}$ onto subspace after subtracting the mean of the latent space to infer the mean, $E[\mathbf{y}|\mathbf{x},\mathbf{T},\mathbf{c}]$:

$$E[\mathbf{y}|\mathbf{x}^{(\mathbf{j})},\mathbf{T},\mathbf{c}] = \mathbf{Q_c}(\mathbf{T}^{'}\mathbf{x}-\mu_\mathbf{c}) \tag{10}$$

This is a K dimensional vector for each $\mathbf{T}$, and can be computed for each dimension (factor) as:

$$E[\mathbf{y_k}|\mathbf{x},\mathbf{T},\mathbf{c}] = \sum_{\mathbf{i}=1}^{N}(\mathbf{Q_c})_{\mathbf{k},\mathbf{i}}(\mathbf{x}(\mathbf{i}-\mathbf{T})-\mu_\mathbf{c}(\mathbf{i})) \tag{11}$$

where the summation over $\mathbf{i}$ for all $\mathbf{T}$ is a correlation (eqn. 5) and thus can be computed for all $\mathbf{T}$ in tandem by:

$$E[\mathbf{y_k}|\mathbf{x},\mathbf{T},\mathbf{c}] = (\mathbf{Q_c})_{\mathbf{k},:}\odot(\mathbf{x}-\mu_\mathbf{c}) \tag{12}$$

The distribution, $p(\mathbf{y}|\mathbf{x},\mathbf{c})=\sum_{\mathbf{T}\in\mathcal{T}}\mathbf{P}(\mathbf{T}|\mathbf{x},\mathbf{c})\mathbf{p}(\mathbf{y}|\mathbf{x},\mathbf{T},\mathbf{c})$ is a mixture of $\mathcal{T}$ Gaussians with mixing proportions, P$(\mathbf{T}|\mathbf{x},\mathbf{c})$ and its mean subspace coordinates can be computed using $|\mathcal{T}|$ additions through:

$$E[\mathbf{y}|\mathbf{x},\mathbf{c}] = \sum_{\mathbf{T}\in\mathcal{T}}\mathbf{P}(\mathbf{T}|\mathbf{x},\mathbf{c})\mathbf{E}[\mathbf{y}|\mathbf{x},\mathbf{T},\mathbf{c}] \tag{13}$$

The interaction between each pair of subspace coordinates can be obtained by using the autocorrelation between them:

$$\begin{aligned} E[(\mathbf{y}\mathbf{y}')_{\mathbf{i},\mathbf{j}}|\mathbf{x},\mathbf{c}] &= \sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}|\mathbf{x},\mathbf{c})\mathbf{E}[\mathbf{y_i}|\mathbf{x},\mathbf{T},\mathbf{c}]\mathbf{E}[\mathbf{y_j}|\mathbf{x},\mathbf{T},\mathbf{c}]' + \sum_{\mathbf{T}\in\mathcal{T}}\mathbf{P}(\mathbf{T}|\mathbf{x},\mathbf{c})\mathtt{COV}[\mathbf{y_{i,j}}|\mathbf{x},\mathbf{T},\mathbf{c}] \\ &= \sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}|\mathbf{x},\mathbf{c})(\mathbf{E}[\mathbf{y_i}|\mathbf{x},\mathbf{T},\mathbf{c}]\circ\mathbf{E}[\mathbf{y_j}|\mathbf{x},\mathbf{T},\mathbf{c}]) + \mathtt{COV}[\mathbf{y_{i,j}}|\mathbf{x},\mathbf{c}] \end{aligned} \tag{14}$$

3) Inferring the correlation between $\mathbf{z}$ *and* $\mathbf{y}$*:* For a given $\mathbf{x}$ and c, the correlation between $\mathbf{y}$ and $\mathbf{z}$ gives the degree of relationship between the latent image $\mathbf{z}$ and its explanation in the linear subspace through the factors $\mathbf{y}$:

$$E[\mathbf{z}\mathbf{y}'|\mathbf{c},\mathbf{x}] = \sum_{\mathbf{T}\in\mathcal{T}} \mathbf{P}(\mathbf{T}|\mathbf{c},\mathbf{x})\mathbf{E}[\mathbf{z}|\mathbf{T},\mathbf{c},\mathbf{x}]\mathbf{E}[\mathbf{y}|\mathbf{T},\mathbf{c},\mathbf{x}]' + \sum_{\mathbf{T}\in\mathcal{T}} \mathtt{COV}[\mathbf{z}|\mathbf{x},\mathbf{y},\mathbf{T},\mathbf{c}]\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c}\mathtt{COV}[\mathbf{y}|\mathbf{x},\mathbf{T},\mathbf{c}]$$

The covariance of the latent factors are first mapped to the input space through the corresponding class-dependent projection matrix, $\mathbf{\Lambda_c}$, and the contribution of each dimension is scaled according to the inverse noise level, $\mathbf{\Phi_c^{-1}}$, before multiplying with covariance in the input space. Naively performing this computation through summing over transformations will require computing and storing $E[\mathbf{z}|\mathbf{T},\mathbf{c},\mathbf{x}]$ for all $\mathbf{T}$. Instead, a little linear algebraic manipulation results in:

$$E[\mathbf{z}\mathbf{y}'|\mathbf{c},\mathbf{x}] = \Big[\mathbf{E}[\mathbf{z}\mathbf{z}'|\mathbf{x},\mathbf{c}]\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c} - \mathbf{E}[\mathbf{z}|\mathbf{x},\mathbf{c}]\mu_\mathbf{c}'\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c}\Big]\Big[\mathbf{\Lambda_c'}\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c} + \mathbf{I}\Big]^{-\mathbf{1}} \tag{15}$$

This form requires storage of at most N×K matrices and in appendix we show how it can be efficiently computed.

4) Inferring the pre-transformation noise: In the generative model, for a given class c, the pre transformation noise $\mathbf{\Phi_c}$ captures the uncertainty between $\mathbf{z}$ and its subspace representation. Given an observation $\mathbf{x}$, we can compute this uncertainty as the expected mean squared deviation between $\mathbf{z}$ and $\mu_\mathbf{c} + \mathbf{\Lambda_c}\mathbf{y}$:

$$\begin{aligned} E[diag\Big((\mathbf{z}-\mu_\mathbf{c}-\mathbf{\Lambda_c}\mathbf{y})(\mathbf{z}-\mu_\mathbf{c}-\mathbf{\Lambda_c}\mathbf{y})'\Big)|\mathbf{x},\mathbf{c}] &= E[diag(\mathbf{z}\mathbf{z}^\mathbf{T}|\mathbf{c},\mathbf{x})] + \mu_\mathbf{c}\circ\mu_\mathbf{c} + \mathbf{2}(\mathbf{\Lambda_c}\mathbf{E}[\mathbf{y}|\mathbf{c},\mathbf{x}])\circ\mu_\mathbf{c} \\ &\quad + \sum_{k=1}^{K}(\mathbf{\Lambda_c})_{:,\mathbf{k}}\circ\Big((\mathbf{\Lambda_c}\mathbf{E}[\mathbf{y}\mathbf{y}^\mathbf{T}|\mathbf{x},\mathbf{c}])_{:,\mathbf{k}} - \mathbf{2}\mathbf{E}[\mathbf{z}\mathbf{y}^\mathbf{T}|\mathbf{c},\mathbf{x}]\Big) - \mathbf{2}\mu_\mathbf{c}\circ\mathbf{E}[\mathbf{z}|\mathbf{c},\mathbf{x}] \end{aligned}$$

where $E[diag(\mathbf{z}\mathbf{z}^\mathbf{T}|\mathbf{c},\mathbf{x})]$ is computed efficiently as described in appendix and rest of the expected values are computed as described in the previous subsections.

5) Inferring the cluster index and transformation: Given an observation, the joint distribution over transformation and class index is obtained by applying the Bayes rule:

$$P(\mathbf{T},\mathbf{c}|\mathbf{x}) = \frac{\mathbf{p}(\mathbf{x}|\mathbf{T},\mathbf{c})\mathbf{P}(\mathbf{T})\mathbf{P}(\mathbf{c})}{\mathbf{p}(\mathbf{x})}$$

Evaluating $p(\mathbf{x}|\mathbf{T},\mathbf{c})$ (eqn. 3) involves computing the determinant of the covariance matrix. Using assumptions $\mathbf{\Psi} = \psi\mathbf{I}$ and $\mathbf{T}'\mathbf{T} = \mathbf{T}\mathbf{T}' = \mathbf{I}$ and properties of determinants, the determinant can be efficiently computed as:

$$\begin{aligned} |\mathtt{COV}[\mathbf{x}|\mathbf{T},\mathbf{c}]| &= |\mathbf{T}(\mathbf{\Lambda_c}\mathbf{\Lambda_c'} + \mathbf{\Phi_c})\mathbf{T}' + \mathbf{\Psi}| = |\mathbf{T}(\mathbf{\Lambda_c}\mathbf{\Lambda_c'} + \mathbf{\Phi_c} + \mathbf{\Psi})\mathbf{T}'| \\ &= |\mathbf{T}\mathbf{T}'||\mathbf{\Lambda_c}\mathbf{\Lambda_c'} + \mathbf{\Phi_c} + \mathbf{\Psi}| = |\mathbf{\Lambda_c'}(\mathbf{\Phi_c}+\mathbf{\Psi})^{-\mathbf{1}}\mathbf{\Lambda_c'} + \mathbf{I}||\mathbf{\Phi_c}+\mathbf{\Psi}| \end{aligned}$$

Defining , $\mathbf{A_c} = (\mathbf{\Phi_c}+\mathbf{\Psi})^{-\mathbf{1}}$, and $\mathbf{B} = \mathtt{COV}[\mathbf{y}|\mathbf{x},\mathbf{c},\mathbf{T}]$, the Mahalanobis distance between observation $\mathbf{x}$ and the latent image can be expressed as:

$$(\mathbf{T}^{'}\mathbf{x})^{'}\mathbf{A_c}\mathbf{T}^{'}\mathbf{x} - \mathbf{2}\mu_\mathbf{c}^{'}\mathbf{A_c}\mathbf{T}^{'}\mathbf{x} + \mu_\mathbf{c}^\mathbf{T}\mathbf{A_c}\mu_\mathbf{c} - \mathbf{E}[\mathbf{y}|\mathbf{T},\mathbf{x}]^{'}\mathbf{B^{-1}}\mathbf{E}[\mathbf{y}|\mathbf{T},\mathbf{x}]$$

The last two terms in the above expression can be computed efficiently (sec. IV-B.2). The first two terms can be computed for all $\mathbf{T}$ in O(NlogN) time by expanding the multiplication and looking at a particular term:

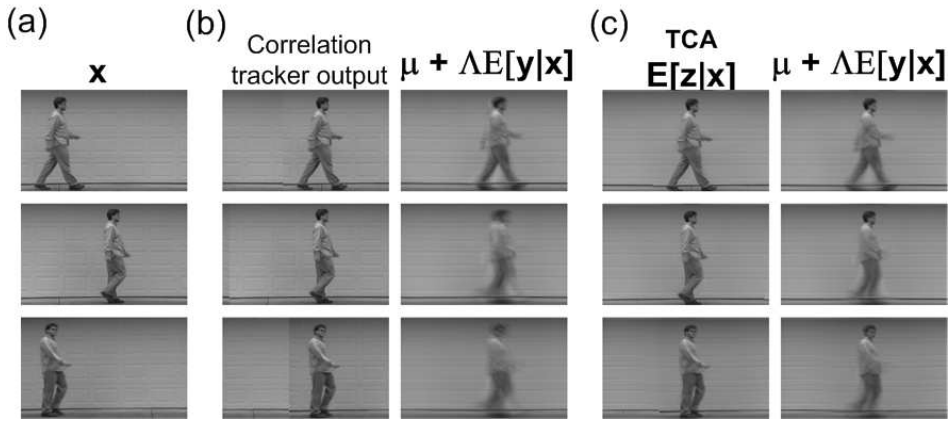

Fig. 3. Comparison of FA applied on data normalized for translations using correlation tracker and TCA. (a)Frames from sequence. (b) shift normalized frames, using correlation-based tracker and $E[\mu + \mathbf{\Lambda y}]$ obtained using factor analysis model. (c)$E[\mathbf{z}|\mathbf{x}]$ and $E[\mu + \mathbf{\Lambda y}]$ for the TCA model.

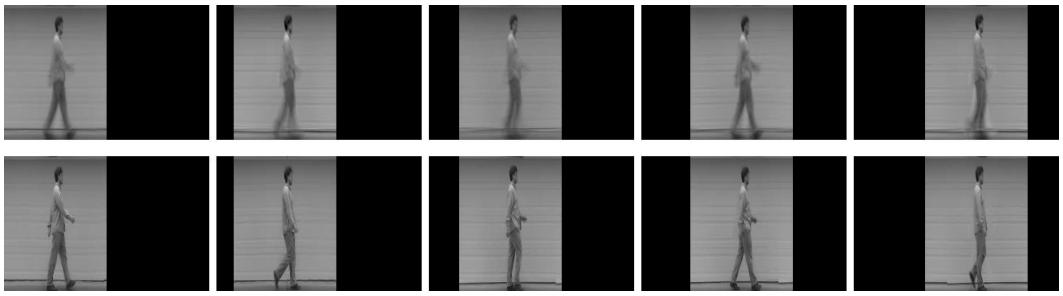

Fig. 4. Simulated walk sequence synthesized after training an AR model on the subspace and image motion parameters. The sequence enlarged for better viewing of translations. The first row contains a few frames from the sequence simulated directly from the model. The second row contains a few frames from the video texture generated by picking the frames in the original sequence for which the recent subspace trajectory was similar to the one generated $\mathbf{y}$ by the AR model.

$$\begin{aligned}(\mathbf{T}^{'}\mathbf{x})^{'}\mathbf{A_c}\mathbf{T}^{'}\mathbf{x} - \mathbf{2}\mu_{\mathbf{c}}^{'}\mathbf{A_c}\mathbf{T}^{'}\mathbf{x} &= \sum_{\mathbf{i}=1}^{N}\mathbf{A_c(i)}(\mathbf{x(i-T)} \circ \mathbf{x(i-T)}) - \mathbf{2}\sum_{\mathbf{i=1}}^{\mathbf{N}}(\mathbf{A_c(i)} \circ \mu_{\mathbf{c}}\mathbf{(i)})\mathbf{x(i-T)} \\ &= (diag(\mathbf{A_c})) \odot (\mathbf{x} \circ \mathbf{x}) - \mathbf{2}(\mathbf{diag}(\mathbf{A_c}) \circ \mu_{\mathbf{c}}) \odot \mathbf{x}\end{aligned}$$

Thus, each term involves correlating the observation with the parameters governing the latent space.

V. Experimental Results

A. Modelling a walking person

Fig. 3a. shows three 165x285 frames from a video sequence of a person walking. To effectively summarize the sequence, we would like to learn a compact representation for the dynamically and periodically changing hand and leg movements.

Regular principal component analysis or factor analysis can not account for translational motion well and thus will learn a representation that focuses more on learning linearized shifts, and less on the more interesting motion of hands and legs (Fig. 1a.) An approach for solving this problem is to track the object using, for example, a correlation tracker and then learn

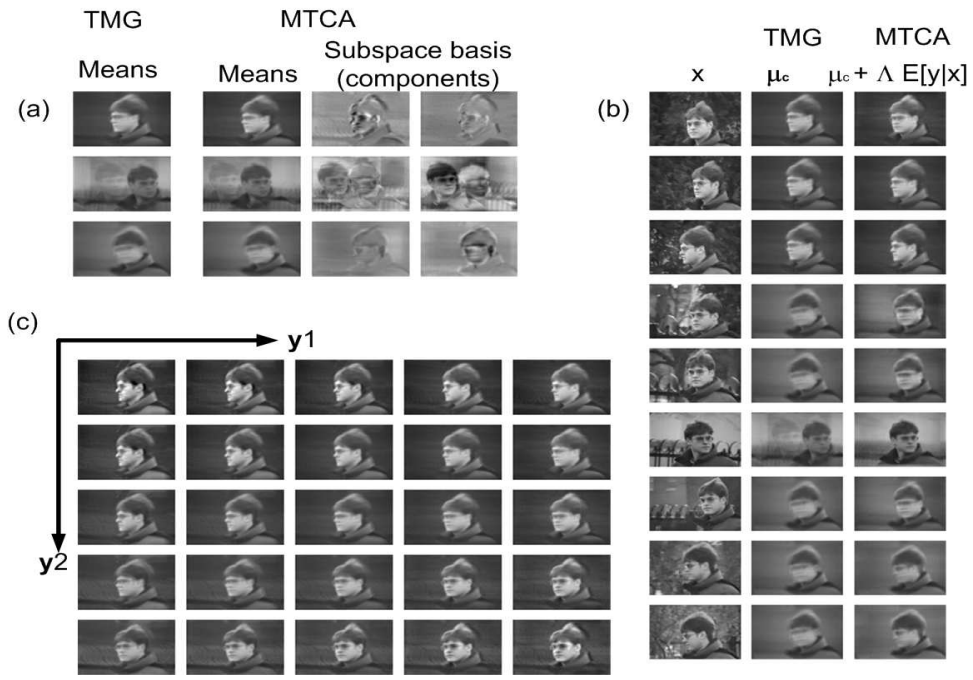

Fig. 5. Transformation invariant clustering with and without a subspace model: (a) Parameters of a three-cluster transformed mixture of Gaussians (TMG) [5], and a three-cluster mixture of transformed component analyzers (MTCA); (b) some of the frames from the video sequence $\mathbf{x}$, corresponding TMG mean $\mu_{\mathbf{c}}$ and the object appearance $\mu_{\mathbf{c}} + \mathbf{\Lambda_c}\mathbf{E}[\mathbf{y}|\mathbf{x}, \mathbf{c}]$ in the corresponding subspace of MTCA; (c) An illustration of the role of components for the first class. Factor $\mathbf{y_1}$ tends to model lighting variation and $\mathbf{y_2}$ tends to model small out-of-plane rotations

a regular factor analysis model on the shift-normalized images. The parameters learned using this approach are shown in Fig. 1b. Without having a good representation for the object being tracked, the tracker fails to provide the perfect tracking necessary for precise subspace modelling of limb motion and thus the inferred subspace projection is blurred. (Fig. 3b.) As TCA performs tracking and learns appearance model jointly, not only does it avoids the tracker initialization that plagues the "tracking first" approaches, but also provides perfectly aligned $E[\mathbf{z}|\mathbf{x}]$ and infers a much cleaner projection $E[\mu + \mathbf{\Lambda y}|\mathbf{x}]$ (Fig. 3c.). The learned TCA model can be used to create video textures based on frame reshuffling similar to [17]. However, instead of shuffling frames based directly on pixel similarity, we use the subspace position $\mathbf{y^{(t)}}$ and image position $\mathbf{T^{(t)}}$ generated from an AR process [16], and for each t find the best frame u in the original video $\mathbf{x^{(u)}}$ for which the window $E[\mathbf{y}|\mathbf{x^{(u)}}], \mathbf{E}[\mathbf{y}|\mathbf{x^{(u-1)}}], ...\mathbf{E}[\mathbf{y}|\mathbf{x^{(u-9)}}]$ is the most similar to $\mathbf{y^{(t)}}, ..., \mathbf{y^{(t-9)}}$. Then, the generated transformation $\mathbf{T}$ is applied on the normalized image $E[\mathbf{z}|\mathbf{x^{(u)}}]$. The result is shown in fig. 4b and contains a bit sharper images than the ones simulated directly from the generative model, fig. 4a. We let the simulated walk last longer than in the original sequence letting TCA live on twice as wide frames.

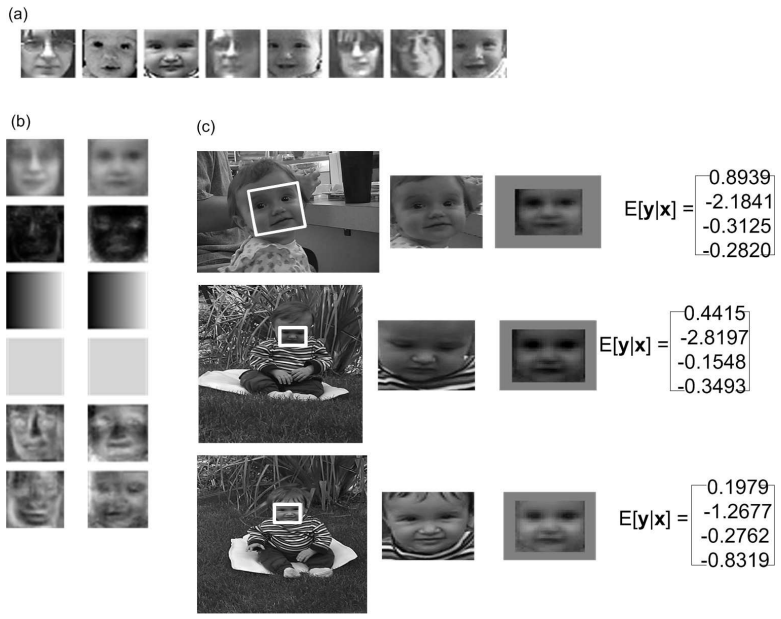

Fig. 6. Clustering faces extracted from a personal database prepared using face detector. (a) Examples from the training set (b) Means, variances and components for two classes learned using MTCA. (c) 1^{st} column contains several photos in which the detector [15] failed to find the face. 2^{nd} column contains central 100x100 portion of $E[\mathbf{z}|\mathbf{x}, \mathbf{c}]$. 3^{rd} column contains central 100x100 portion of $\mu_{\mathbf{c}} + \mathbf{\Lambda_c}\mathbf{E}[\mathbf{y}|\mathbf{x}, \mathbf{c}]$.

B. Clustering face poses

Here, the goal is to cluster images rom a video sequence of a person with different facial poses walking across cluttered background. The first column of fig. 5b shows representative examples.

We first trained a transformation-invariant mixture of Gaussians [6] with 3 clusters. Fig. 5a shows that the means learned using TMG captures the three generic poses in the data. However, this model is inadequate to capture small variations in lighting and small out-of-plane rotations without many more classes. Hence, it uses the pre- and post- transformation noise to model these variations.

We trained a mixture of transformation-invariant component analysis model with 3 classes and 2 factors, initializing the parameters to those learned by TMG. Fig. 5a compares the parameters learned using TMG and MTCA. As seen, the MTCA model learns to capture small variations around the cluster means. For example, for the first cluster, the two subspace coordinates tend to model out-of-plane rotations and illumination changes (Fig. 5c). In Fig. 5b, we compare the mean latent image $E[\mu_{\mathbf{c}} + \mathbf{\Lambda_c}\mathbf{y}|\mathbf{x}]$, ($c = argmax_c P(c|\mathbf{x})$), of TMG and MTCA for various training examples, thereby illustrating better tracking and appearance modelling of MTCA.

C. Clustering and face recognition

We used a standard face detector [15] to obtain 85 32x32 images of faces of 2 persons, from a personal photo database of a mother and her daughter. In fig. 6a. we present examples from the training set.

We learned a MTCA model with 2 classes and 4 factors. To model global lighting variation, we preset one of the factors to be uniform at .01 (see fig. 6b.). This handles linearized version of the ambient lighting condition. We also preset another factor to a smoothly varying brightness image to capture side illumination (see fig. 6b.). The other two components are learned and they model slight appearance deformation such as facial expressions. The model learned to cluster faces in the training set with 94.12% accuracy.

An interesting application of this model is to use the learned representation of the faces to detect and recognize faces in the original photos. For instance, the face detector did not recognize faces in many photographs, three of which are shown in fig. 6c. Using the model we learned, we were able to recognize them (fig. 6c). For this, we increased the resolution of model parameters $\{\mu_{\mathbf{c}}, \mathbf{\Lambda_c}, \mathbf{\Phi_c}\}$ to match the resolution of photos (640x480), padding around the original parameters with uniform mean, zero factors and high variance. With the expanded parameters, we performed inference, inferring most likely class,c, most likely transformation,$\mathbf{T}$ for that class and $E[\mathbf{z}|\mathbf{x}, \mathbf{c}]$. In addition to all possible shifts, we incorporated 3 rotations and 4 scales as possible transformations. In fig. 6c, we present three examples which were not in the training set and the face detector we used failed. In these three cases MTCA detected and *recognized* the face correctly as belonging to class 2.

VI. Conclusions

Transformation-invariant component analyzers is a technique for performing linear dimensionality reduction, invariant to a set of global transformations. Here, we have described an efficient EM algorithm for learning this model through effective use of identities from matrix algebra and the use of fast Fourier transforms. The generative model-based approach provides us the ability to extend the model. In this work, we also presented the mixture of TCA that clusters data while performing TCA. In this paper, we have illustrated the use of this technique in a number of applications using visual data. We believe that this tremendous speed up makes the model practical for a number of other vision applications and different kinds of data. Further experimental results and MATLAB software can be obtained from http://www.psi.utoronto.ca.

VII. Appendix: More details

For the update of $\mathbf{\Phi_c}$, we require to compute the following term:

$$\begin{aligned}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x}, \mathbf{c}) \mathbf{diag}\Big(\mathbf{E}[\mathbf{z}|\mathbf{T}, \mathbf{x}, \mathbf{c}]\mathbf{E}[\mathbf{z}|\mathbf{T}, \mathbf{x}, \mathbf{c}]^{'}\Big) &= \sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x}, \mathbf{c}) \Big(\mathbf{E}[\mathbf{z}|\mathbf{T}, \mathbf{x}, \mathbf{c}] \circ \mathbf{E}[\mathbf{z}|\mathbf{T}, \mathbf{x}, \mathbf{c}]\Big) \\ &= \sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x}) \Big((\mathbf{h} + \mathbf{g_T}) \circ (\mathbf{h} + \mathbf{g_T})\Big)\end{aligned}$$

where, $\mathbf{h} = \mathsf{COV}[\mathbf{z}|\mathbf{T},\mathbf{x}](\mathbf{\Lambda_c}\mathbf{\Lambda_c^T} + \mathbf{\Phi_c})^{-1}\mu_\mathbf{c}$ is independent of $\mathbf{T}$ and can be directly computed as described before and $\mathbf{g_T} = \mathsf{COV}[\mathbf{z}|\mathbf{T},\mathbf{x},\mathbf{c}]\mathbf{\Psi^{-1}}\mathbf{T}^{'}\mathbf{x}$. So,

$$\begin{aligned}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x},\mathbf{c})\mathsf{diag}\Big(\mathbf{E}[\mathbf{z}|\mathbf{T},\mathbf{x},\mathbf{c}]\mathbf{E}[\mathbf{z}|\mathbf{T},\mathbf{x},\mathbf{c}]^{'}\Big) &= \mathbf{h}\circ\mathbf{h} + 2\mathbf{h}\circ\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\mathbf{g_T} + \sum_{\mathbf{T}}\mathbf{P}(\mathbf{T}|\mathbf{x})(\mathbf{g_T}\circ\mathbf{g_T}) \\ &= \mathbf{h}\circ\mathbf{h} + 2\mathbf{h}\circ\mathsf{COV}[\mathbf{z}|\mathbf{T},\mathbf{x}]\mathbf{\Psi^{-1}}\sum_{\mathbf{T}}\mathbf{P}(\mathbf{T}|\mathbf{x})\mathbf{T}^{'}\mathbf{x} + \sum_{\mathbf{T}}\mathbf{P}(\mathbf{T}|\mathbf{x})(\mathbf{g_T}\circ\mathbf{g_T})\end{aligned}$$

The first term can be directly computed. To compute the second term, we first compute $\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x},\mathbf{c})\mathbf{T}^{'}\mathbf{x}$ using FFTs in order $N\log N$ and then compute $\mathsf{COV}[\mathbf{z}|\mathbf{T},\mathbf{x},\mathbf{c}]\mathbf{\Psi^{-1}}\sum_{\mathbf{T}}\mathbf{P}(\mathbf{T}|\mathbf{x},\mathbf{c})\mathbf{T}^{'}\mathbf{x}$ and then the entire term. The third term is:

$$\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x},\mathbf{c})(\mathbf{g_T}\circ\mathbf{g_T}) = \sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x},\mathbf{c})\Big((\mathsf{COV}[\mathbf{z}|\mathbf{T},\mathbf{x},\mathbf{c}]\mathbf{\Psi^{-1}}\mathbf{T}^{'}\mathbf{x})\circ(\mathsf{COV}[\mathbf{z}|\mathbf{T},\mathbf{x},\mathbf{c}]\mathbf{\Psi^{-1}}\mathbf{T}^{'}\mathbf{x})\Big) \quad (16)$$

Using expansion of $\mathsf{COV}[\mathbf{z}|\mathbf{T},\mathbf{x},\mathbf{c}]$, this simplifies to

$\mathbf{D^2\Psi^{-2}}\Big(\sum_{\mathbf{T}}\mathbf{P}(\mathbf{T}|\mathbf{x},\mathbf{c})\mathsf{diag}(\mathbf{T}^{'}\mathbf{x}\mathbf{x}^{'}\mathbf{T})\Big) + \mathbf{2D\Psi^{-1}}\sum_{\mathbf{T}}\mathbf{P}(\mathbf{T}|\mathbf{x})\mathsf{diag}(\mathbf{EFT}^{'}\mathbf{x}\mathbf{x}^{'}\mathbf{T}) + \sum_{\mathbf{T}}\mathbf{P}(\mathbf{T}|\mathbf{x})\mathsf{diag}(\mathbf{EFT}^{'}\mathbf{x}\mathbf{x^T}\mathbf{TE}^{'}\mathbf{F}^{'})$

To compute the first term, we first compute the terms inside the summation using FFTs by observing that an element in vector that results from the summation is $\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\mathbf{x}(\mathbf{i}-\mathbf{T})^{\mathbf{2}}$. This can be computed for all $\mathbf{i}$ using FFTs in $N\log N$. For the second term, the summation is calculated as follows: $\mathbf{i}^{\mathsf{th}}$ element in $\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\mathsf{diag}(\mathbf{EFT^T}\mathbf{x}\mathbf{x^T}\mathbf{T})$ is

$$\begin{aligned}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\sum_{\mathbf{j,k}}\mathbf{E_{i,j}}\mathbf{F_{j,k}}\mathbf{x}(\mathbf{k}-\mathbf{T})\mathbf{x}(\mathbf{i}-\mathbf{T}) &= \sum_{j=1:K}\mathbf{E}_{i,j}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\mathbf{x}(\mathbf{i}-\mathbf{T})\sum_{\mathbf{k=1:N}}\mathbf{F_{j,k}}\mathbf{x}(\mathbf{k}-\mathbf{T}) \\ &= \sum_{j=1:K}\mathbf{E}_{i,j}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\mathbf{a}(\mathbf{T},\mathbf{j})\mathbf{x}(\mathbf{i}-\mathbf{T}) \\ &= \sum_{j=1:K}\mathbf{E}_{i,j}\mathbf{C}(i,j)\end{aligned}$$

$\mathbf{a}(\mathbf{T},\mathbf{j}) = \sum_{\mathbf{k=1:N}}\mathbf{F_{j,k}}\mathbf{x}(\mathbf{k}-\mathbf{T})$ can be computed for all j and all $\mathbf{T}$s in K FFTs, taking a total of $KN\log N$ operations. Similarly, $\mathbf{C}(i,j)$ can be calculated. To compute the third term, we need to first compute the summation. This can be done by considering $\mathbf{i}^{th}$ element in the summation, which is given by:

$$\begin{aligned}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\Big(\sum_{\mathbf{j,k}}\mathbf{E_{i,j}}\mathbf{F_{j,k}}\mathbf{x}(\mathbf{k}-\mathbf{T})\Big)^{\mathbf{2}} &= \sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\sum_{\mathbf{j,k}}\mathbf{P_{i,j}}\mathbf{F_{j,k}}\mathbf{x}(\mathbf{k}-\mathbf{T})\sum_{\mathbf{j',k'}}\mathbf{P_{i,j'}}\mathbf{F_{j',k'}}\mathbf{x}(\mathbf{k}^{'}-\mathbf{T}) \\ &= \sum_{j}\mathbf{E}_{i,j}\sum_{j'}\mathbf{F}_{i,j'}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\sum_{\mathbf{k}}\mathbf{F_{j,k}}\mathbf{x}(\mathbf{k}-\mathbf{T})\sum_{\mathbf{k'}}\mathbf{F_{j',k'}}\mathbf{x}(\mathbf{k}^{'}-\mathbf{T}) \\ &= \sum_{j}\mathbf{E}_{i,j}\sum_{j'}\mathbf{E}_{i,j'}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})\mathbf{a}(\mathbf{T},\mathbf{j})\mathbf{a}(\mathbf{T},\mathbf{j}^{'}) \\ &= \sum_{j}\mathbf{E}_{i,j}\sum_{j'}\mathbf{E}_{i,j'}\mathbf{b}(j,j^{'})\end{aligned}$$

$\mathbf{a}(\mathbf{T},\mathbf{j})$ is computed as suggested above. $\mathbf{b}(j,j^{'})$ is obtained by direct summation. The remaining summations are are also straightforward.

To compute correlation between $\mathbf{z}$ and $\mathbf{y}$ (eqn. 15), we require to compute

$$\begin{aligned}E[\mathbf{z}\mathbf{z}'|\mathbf{x},\mathbf{c}]\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c} &= \sum_{\mathbf{T}\in\mathcal{T}}\Big(E[\mathbf{z}|\mathbf{x},\mathbf{T},\mathbf{c}]\mathbf{E}[\mathbf{z}|\mathbf{x},\mathbf{T},\mathbf{c}]'\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c}\Big) + \mathbf{COV}[\mathbf{z}|\mathbf{x},\mathbf{T},\mathbf{c}]\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c} \\ &= \mathbf{e}_c\sum_{k=1}^{K} P(\mathbf{T}|\mathbf{x},\mathbf{c})(\mathbf{L_{T,c}})_{:,\mathbf{k}} + \mathbf{COV}[\mathbf{z}|\mathbf{x},\mathbf{c}]\mathbf{\Psi^{-1}}\Big(\sum_{\mathbf{k=1}}^{\mathbf{K}}\mathbf{P}(\mathbf{T}|\mathbf{x},\mathbf{c})(\mathbf{L_{T,c}})_{:,\mathbf{k}}\otimes\mathbf{x}\Big) + \mathbf{COV}[\mathbf{z}|\mathbf{x},\mathbf{c}]\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c}\end{aligned}$$

where $\mathbf{e}_c = COV[\mathbf{z}|\mathbf{x},\mathbf{c}](\mathbf{\Lambda_c}\mathbf{\Lambda}_\mathbf{c}' + \mathbf{\Phi_c})^{-\mathbf{1}}\mu_\mathbf{c}$ and $\mathbf{L} = E[\mathbf{z}|\mathbf{x},\mathbf{T},\mathbf{c}]'\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c}$ is a K vector for each transformation $\mathbf{T}$. For all $\mathbf{T}$, each of the K elements of $\mathbf{L}$ can be computed as:

$$(\mathbf{L_{T,c}})_{:,k} = \mathbf{e}_c^{'}\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c} + \mathbf{\Psi^{-1}}\Big((\mathbf{H_c})_{:,\mathbf{k}} \odot \mathbf{x}\Big)$$

where $(\mathbf{H}_c)_{:,k}$ is the k^{th} column of $COV[\mathbf{z}|\mathbf{x},\mathbf{c}]\mathbf{\Phi_c^{-1}}\mathbf{\Lambda_c}$

REFERENCES

[1] M.J. Black, D.J. Fleet, and Y. Yacoob. A framework for modeling appearance change in image sequences. In *Proceedings of international conference on computer vision*, pages 660–667, 1998.

[2] M.J. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proceedings of European conference on computer vision*, pages 329–342, 1996.

[3] D. Chudova, S. Gaffney, and P. Smyth. Probabilistic models for joint clustering and time-warping of multidimensional curves. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 134–141, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

[4] B.S. Everitt. *An introduction to latent variable models*. Chapman and Hall, NY, 1982.

[5] B. Frey and N. Jojic. Learning mixture models of images and inferring spatial transformations using the em algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[6] B. Frey and N. Jojic. Transformed component analysis: Joint estimation of spatial transformations and image components. In *Proceedings of the international conference on computer vision*, 1999.

[7] B. Frey and N. Jojic. Fast, large-scale transformation-invariant clustering. In *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, 2002.

[8] B.J. Frey, A. Colmenarez, and T.S Huang. Mixtures of local linear subspaces for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[9] B.J. Frey and N. Jojic. Transformation-invariant clustering using the em algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):1–17, 2003.

[10] B.J. Frey, N. Jojic, and A. Kannan. Layered density models and unsupervised video analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[11] Z. Ghahramani and G. Hinton. The em algorithm for mixtures of factor analyzers. In *University of Toronto Technical Report*, 1996.

[12] G. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8:65–74, 1997.

[13] I.T. Jolliffe. *Principal component analysis*. Springer-Verlag, NY, 1986.

[14] A. Kannan, N. Jojic, and B. Frey. Fast transformation-invariant factor analysis. In *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, 2003.

[15] S.Z. Li, L. Zhu, Z.Q. Zhang, and H.J. Zhang. Learning to detect multi-view faces in real-time. In *Proceedings of the 2nd International Conference on Development and Learning*, IEEE Computer Society, Washington, DC, USA, 2002.

[16] A. Neumaier and T. Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Math Software*, 27(1):27–57, 2001.

[17] A. Schdl, R. Szeliski, D. Salesin, and I. Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, ACM Press, New York, NY, USA, 2000.

[18] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[19] G. Wolberg and S. Zokai. Robust image registration using log-polar transform. In *Proceedings IEEE international conference on image processing*, pages 493–496, 2000.