Transformation-Invariant Clustering Using the EM Algorithm

Brendan J. Frey, *Member*, *IEEE Computer Society*, and Nebojsa Jojic, *Member*, *IEEE Computer Society*

Abstract—Clustering is a simple, effective way to derive useful representations of data, such as images and videos. Clustering explains the input as one of several prototypes, plus noise. In situations where each input has been randomly transformed (e.g., by translation, rotation, and shearing in images and videos), clustering techniques tend to extract cluster centers that account for variations in the input due to transformations, instead of more interesting and potentially useful structure. For example, if images from a video sequence of a person walking across a cluttered background are clustered, it would be more useful for the different clusters to represent different poses and expressions, instead of different positions of the person and different configurations of the background clutter. We describe a way to add transformation invariance to mixture models, by approximating the nonlinear transformation manifold by a discrete set of points. We show how the expectation maximization algorithm can be used to jointly learn clusters, while at the same time inferring the transformation associated with each input. We compare this technique with other methods for filtering noisy images obtained from a scanning electron microscope, clustering images from videos of faces into different categories of identification and pose and removing foreground obstructions from video. We also demonstrate that the new technique is quite insensitive to initial conditions and works better than standard techniques, even when the standard techniques are provided with extra data.

Index Terms—Generative models, transformation, transformation-invariance, clustering, video summary, filtering, EM algorithm, probability model.

1 Introduction

Our approach to solving problems in computer vision is to formulate them as problems of probabilistic and statistical inference in complex, parameterized probability models. A batch of images or a video sequence is used to learn a probable set of model parameters. Then, each input image is interpreted by inferring the distributions over hidden variables, such as "object class," "object position," etc. Also, the model parameters can be used to summarize the data—e.g., render prototypical appearances of the objects. By carefully specifying the structure of the probability models (i.e., conditional independencies between variables and parameters) and by inventing efficient inference algorithms, we hope to solve general classes of computer vision problems.

We present a technique for automatically learning models of different types of object from unlabeled images that include background clutter and spatial transformations, such as translation, rotation and shearing.

For example, Fig. 2a shows several 140×56 gray-scale images obtained from a scanning electron microscope. The electron detectors and the high-speed electrical circuits randomly translate the images and add noise [14]. Standard filtering techniques are not appropriate here, since the images are not aligned. As shown later in this paper, the images cannot be properly aligned using correlation because of the high level of noise. Our technique is able to automatically align the images and estimate the appearance of the aligned images, using a maximum-likelihood criterion.

Fig. 4a shows several 88×56 gray-scale head-and-shoulder images of a person walking outdoors. The video camera did not track the person's head perfectly, so the head appears at different locations in the frame. The images include variation in the pose of the head, as well as background clutter—some of which appears in multiple images. Aligning the images without a model of the person's appearance is difficult. Even when the images are treated as a video sequence, standard blob-tracking methods do not work well due to the presence of coherent background clutter (such as the tree and the gate). Our technique is able to automatically align the images, despite sensor noise and background clutter, and robustly learn a mixture of pose-based appearance maps.

We propose a general purpose statistical method that can jointly normalize for transformations that occur in the training data, and learn a maximum-likelihood density model of the normalized data [9], [10]. The technique can be applied to video sequences, but does not require that the images be temporally ordered. Improvements in performance can be achieved by introducing temporal dependencies, as described in [19], [12].

One approach to predicting the transformation in an input image is to provide a training set of images plus their transformations to a supervised learning algorithm, such as classification and regression trees, multilayer perceptrons, Gaussian process regression, support vector classifiers, nearest-neighborhood methods (which may operate in linear subspaces spanned by eigen-vectors [25], [22]) and adaptive-metric nearest-neighbor methods [21].

The very different approach we take here is to use *unlabeled data* to train a probability density model of the data (or *generative model*). The goal of generative modeling is to learn a probability model that gives high probability to patterns that

- *B.J. Frey is with the Electrical and Computer Engineering Department, University of Toronto, 10 King's College Road, Toronto, ON M5S 3G4. E-mail: frey@psi.toronto.edu.*
- *N. Jojic is with the Vision Technology Group, Microsoft Research, One Microsoft Way, Redmond, WA 98052.*

Manuscript received 17 Nov. 2000; revised 29 Apr. 2002; accepted 5 May 2002. Recommended for acceptance by W. Freeman.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 113169.

are typical in the training data. Since input/output pairs are not explicitly provided during training, learning algorithms of this sort are called *unsupervised* or *self-supervised*.

Clustering techniques can be viewed as maximum-likelihood parameter estimation of a mixture model, where the density in the input space is given by a sum of unimodal densities, each of which corresponds to a cluster. By restricting mixture models in various ways, maximum likelihood estimation corresponds to standard nonprobabilistic algorithms. For example, if the densities in the mixture are Gaussians and the covariance matrices of the Gaussians are set to $\epsilon\mathbf{I}$, $\epsilon \rightarrow 0$, maximum-likelihood estimation simplifies to k-means clustering.

One important advantage of the probabilistic versions of clustering is that the probability model explicitly contains random variables that account for different types of noise. For example, suppose the training data consists of pictures of different views of a centered object, with noisy background clutter. If the variance in light intensity for the pixels containing background clutter is larger than the variance for the pixels containing parts of the object, k-means clustering will capture different configurations of the background clutter, instead of the object. In contrast, a mixture of Gaussians can separate the background clutter from the object and explain the clutter as noise.

Unsupervised learning is useful for summarizing data (e.g., learning two representative head-and-shoulder shots from the images in Fig. 4a), filtering data (e.g., denoising the images in Fig. 2a), estimating density models used for data compression, and as a preprocessing step for supervised methods (e.g., removing the shearing from images of handwritten digits before training a classifier in a supervised fashion).

By thinking of unsupervised learning as maximum-likelihood estimation of a density model of the data, we can incorporate extra knowledge about the problem. One way to do this is to include extra *latent variables* (unobserved variables) in the model. The model we present extends the mixture of Gaussians to include "transformation" as a latent variable. The model can be trained in a maximum-likelihood fashion.

In the next section, we describe computationally efficient approaches to modeling transformations. Then, in Section 3, we describe how large transformations in the input can be accounted for by incorporating a discrete "transformation variable" into a Gaussian mixture model, producing a "transformation-invariant mixture of Gaussians" (TMG).

In Section 3.3, we describe how TMG can be fit to a training set using the expectation maximization (EM) algorithm. The learning algorithm uses the current appearance estimate to automatically remove the transformations from the training cases before adjusting the model parameters.

We demonstrate the general usefulness of learning transformation-invariant models through a series of experiments in Section 4. We show that our method is quite insensitive to initial conditions. We also show that our method learns faster and obtains lower errors than standard mixture modeling, even when the latter is given extra training data. We conclude with some remarks about the general usefulness of TMG for clustering and how it relates to other approaches.

Implementation details are important for these models, since a correct implementation can lead, e.g., to a speed-up by a factor of 8×10^{13} for translation-invariant clustering of 320×240 images. In Appendix B, we give the details needed for fast implementations.

In the companion paper entitled "Transformation-Invariant Factor Analysis Using the EM Algorithm," we describe how linear subspace modeling (akin to PCA) can be performed in a way that is invariant to transformations in the input.

2 Approximating the Transformation Manifold

To make data models invariant to a known type of transformation in the input, we would like to make all transformed versions of a particular input "equivalent." Suppose an N-element input undergoes a transformation with 1 degree of freedom—for example, an N-pixel gray-scale image undergoes translation in the x-direction, with wrap-around. Imagine what happens to the point in the N-dimensional pixel intensity space while the object is translated. Due to pixel mixing, a very small amount of subpixel translation will move the point only slightly, so translation will trace a continuous one-dimensional curve in the space of pixel intensities. As illustrated in Fig. 1a, extensive levels of translation will produce a highly nonlinear curve, although the curve can be approximated by a straight line locally. (For example, as a small white dot on a black background is translated, it will activate and then deactivate pixels in sequence.) If K types of continuous transformation are applied, the manifold will be K-dimensional.

Linear approximations of the transformation manifold have been used to significantly improve the performance of supervised classifiers such as nearest neighbors and multilayer perceptrons [23]. Linear generative models (factor analysis, mixtures of factor analysis) have also been modified using linear approximations of the transformation manifold to build in some degree of invariance to transformations [16].

In general, the linear approximation is accurate for small transformations, but is inaccurate for large transformations. In some cases, a multiresolution version of the linear approximation can be used [26], but this approach relies on assumptions about the size of the objects in the images.

For significant levels of transformation, the nonlinear manifold can be better modeled using a discrete approximation. For example, the curve in Fig. 1a can be represented by a set of points (filled discs). In this approach, a discrete set of possible transformations is specified beforehand and parameters are learned so that the model is invariant to the set of transformations. This approach has been used in the supervised framework to design "convolutional neural networks" that are trained using labeled data [20]. We show how this approach can be used for unsupervised learning.

3 Transformation-Invariant Clustering

In this section, we show how to incorporate the discrete approximation described above into generative models for clustering. For now, we assume the model parameters (e.g., cluster centers) are known—in the next section, we show how to estimate them in an unsupervised fashion.

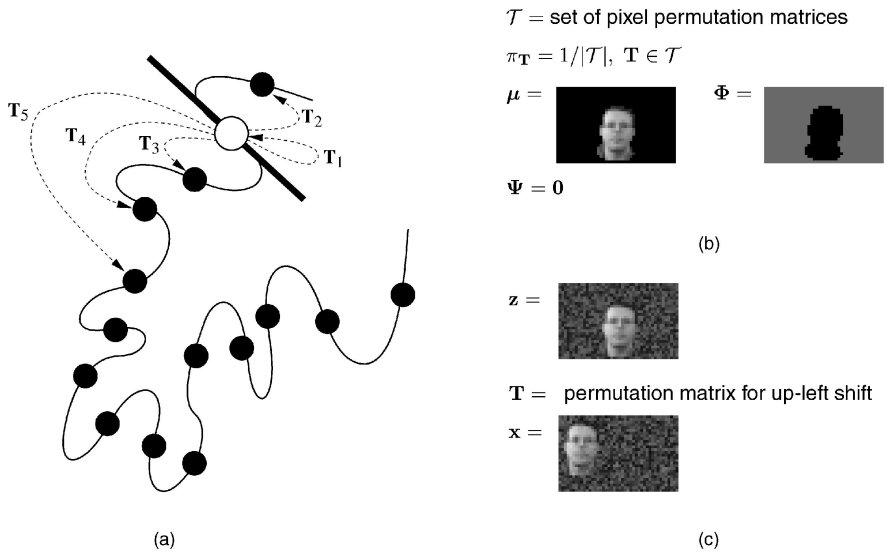


Fig. 1. (a) An N-element input vector is represented by a point (unfilled disc) in an N-dimensional space. When the input undergoes a continuous transformation with 1 degree of freedom, a 1-dimensional manifold is traced. For transformation-invariant data modeling, we want all inputs on this manifold to be equivalent in some sense. Locally, the curve is linear, but high levels of transformation may produce a highly nonlinear curve. We approximate the manifold by discrete points (filled discs) that are obtained by multiplying the original N-vector by a sparse transformation matrix $\mathbf{T}$, from a set of matrices $\mathcal{T}$. (b) Parameters of a transformed Gaussian. A handcrafted model illustrates how a discrete transformation variable is incorporated into a Gaussian model. The mean appearance vector $\boldsymbol{\mu}$ and diagonal covariance matrix $\boldsymbol{\Phi}$ for the latent image vector $\mathbf{z}$ are shown as raster-scanned images. In the case of the variances, black corresponds to low variance, whereas white corresponds to high variance. Whereas $\boldsymbol{\Phi}$ models Gaussian noise that is added to the latent image before the transformation is applied, $\boldsymbol{\Psi}$ models Gaussian noise that is added after the transformation is applied, to obtain the observed image $\mathbf{x}$. (c) Generating from a transformed Gaussian. Values of $\mathbf{z}$, $\mathbf{T}$, and $\mathbf{x}$, sampled from the transformed Gaussian model.

Conditioning on the discrete variables, all of the models presented here are jointly Gaussian, so inference is computationally efficient. Although many expressions may look complicated, they are "straightforward" linear algebra.

See Appendix A for derivations of linear algebra for Gaussian forms that are used throughout this paper.

We represent the discrete set of possible transformations by a set of *sparse* transformation matrices, $\mathcal{T}$. Each matrix $\mathbf{T} \in \mathcal{T}$ in this set is a *sparse* $N \times M$ matrix that operates on an M-vector of untransformed (latent) image pixel intensities to produce an N-vector of transformed, observed image pixel intensities. *Although the transformation matrix representation is used to derive the inference and learning algorithms, the transformation matrices need not be explicitly computed or stored in practice.*

The algorithms we present in this paper properly handle transformations that are not uniquely reversible. For example, it is often useful for the observed image to be a window into a larger latent image. In this case, $M > N$ and the transformation matrices clearly do not have inverses.

Many useful types of transformation can be represented by sparse transformation matrices. For example, rotation and blurring can be represented by matrices that have a small number of nonzero elements per row (e.g., at most six for rotations). Alternatively, these transformations can be approximated by permutation matrices.

Integer-pixel translations of an image can be represented by transformation matrices that have a single nonzero entry in each row. If $M = N$, each matrix is a permutation matrix. For example, integer pixel translations with wrap-around in a 2×3 observed image and a 2×3 latent image are represented by six, 6×6 permutation matrices. Assuming the image is read into vector form in raster-scan order, the set of transformation matrices $\mathcal{T}$ is

$$\left\{ \begin{pmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1 \end{pmatrix}, \begin{pmatrix} 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1\\ 0&0&0&1&0&0 \end{pmatrix}, \right.$$

$$\begin{pmatrix} 0&0&1&0&0&0\\ 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&0&0&0&1\\ 0&0&0&1&0&0\\ 0&0&0&0&1&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1\\ 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&1&0&0&0 \end{pmatrix},$$

$$\left. \begin{pmatrix} 0&0&0&0&1&0\\ 0&0&0&0&0&1\\ 0&0&0&1&0&0\\ 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 1&0&0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0&0&1\\ 0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&1&0&0&0\\ 1&0&0&0&0&0\\ 0&1&0&0&0&0 \end{pmatrix} \right\}.$$

Generally, for an N-pixel latent image and an N-pixel observed image with wrap-around, N transformation matrices are needed to represent all horizontal and vertical translations. For a 320×240 image, $N = 76,800$, and there are $76,800$, $76,800 \times 76,800$ transformation matrices. As mentioned above, *these transformation matrices need not be explicitly computed or stored in practice.*

In fact, the fast Fourier transform technique described in Appendix B can be used to make the algorithms run very fast.

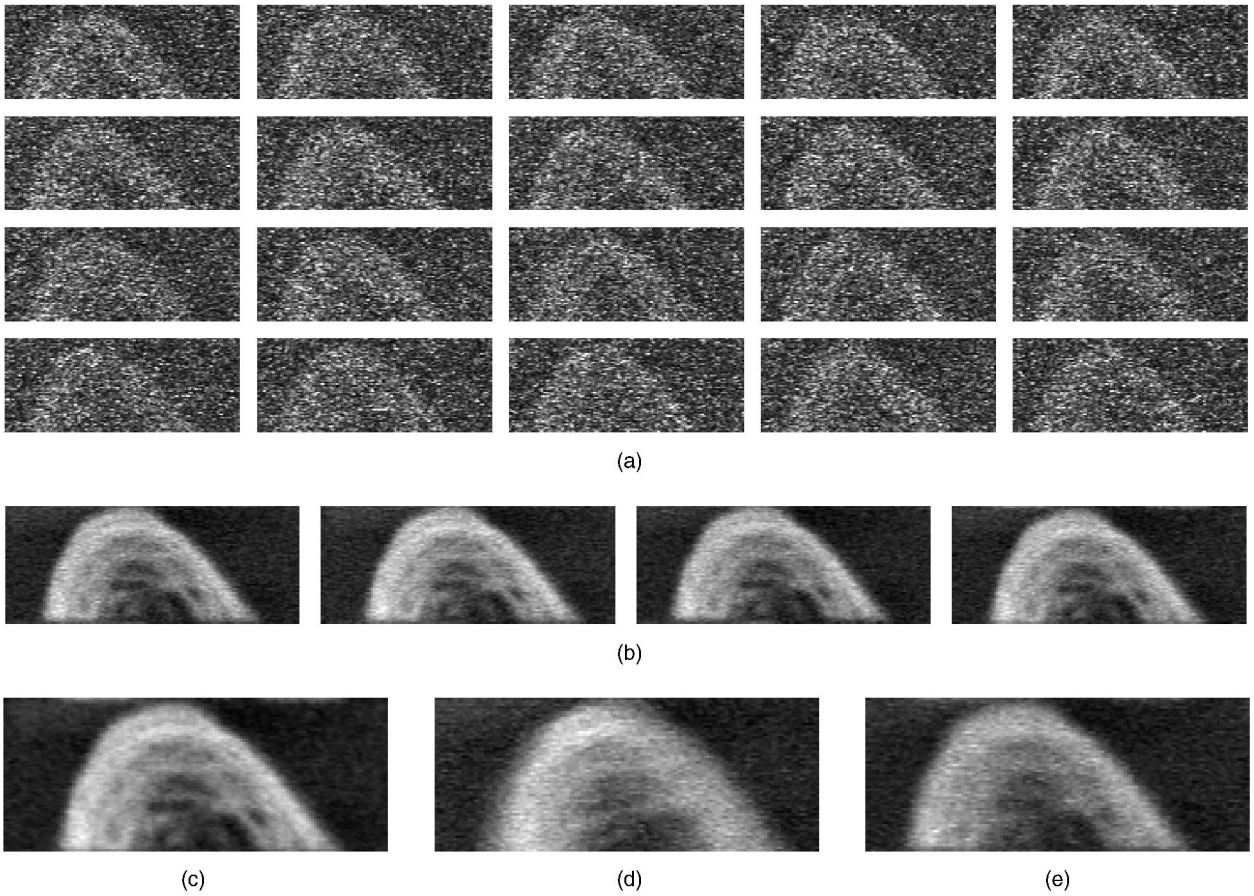


Fig. 2. (a) Some examples from a set of 230, 140×56 images obtained by a scanning electron microscope. The electron detectors and high-speed electrical circuits introduce random translations and noise. (b) The means found after four runs of 50 iterations of EM in a TMG with one cluster, where in each run the initial parameters are selected using a different random seed. The set of transformations is all 7,840 translations with wrap-around. (c) The average of the four mean images shown in (b). (d) The straight average of the 230 training images. (e) The average of the 230 images after each image is aligned to the first one using a greedy correlation measure.

Given the nontransformed *latent image* $\mathbf{z}$ and the transformation $\mathbf{T} \in \mathcal{T}$, we assume the density of the observed image $\mathbf{x}$ is

$$p(\mathbf{x}|\mathbf{T}, \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{T}\mathbf{z}, \mathbf{\Psi}),$$

where $\mathbf{\Psi}$ is a diagonal matrix of sensor noise variances. $\mathcal{N}()$ is the Gaussian density function:

$$\mathcal{N}(\mathbf{y}; \mathbf{m}, \mathbf{S}) = |2\pi\mathbf{S}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{y}-\mathbf{m})^\top \mathbf{S}^{-1}(\mathbf{y}-\mathbf{m})\right],$$

where $\mathbf{m}$ and $\mathbf{S}$ are the mean and covariance of $\mathbf{y}$, and "$^\top$" indicates matrix transpose.

It is sometimes advantageous to set the diagonal matrix of sensor noise variances to zero, $\mathbf{\Psi} = 0$, as described below.

We assume the choice of transformation is independent of the latent image, so

$$p(\mathbf{x}, \mathbf{z}|\mathbf{T}) = p(\mathbf{x}|\mathbf{T}, \mathbf{z})p(\mathbf{z}|\mathbf{T}) = \mathcal{N}(\mathbf{x}; \mathbf{T}\mathbf{z}, \mathbf{\Psi})p(\mathbf{z}). \quad (1)$$

The density of the latent image $p(\mathbf{z})$ can take on various forms as described in the following sections.

We denote the probability of transformation $\mathbf{T}$ by parameter $\pi_{\mathbf{T}}$. Indexing the parameter π by a matrix is a notational convenience—in practice, we associate an integer index with each transformation matrix, so that the ℓth parameter π_ℓ is the probability of the ℓth transformation matrix. The joint distribution over the latent image $\mathbf{z}$, the transformation $\mathbf{T}$ and the observed image $\mathbf{x}$ is

$$p(\mathbf{x}, \mathbf{T}, \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{T}\mathbf{z}, \mathbf{\Psi})p(\mathbf{z})\pi_{\mathbf{T}}. \quad (2)$$

In the following two sections, we show how the density of the latent image, $p(\mathbf{z})$, can be modeled using a Gaussian and a mixture of Gaussians. In the companion paper "Transformation-Invariant Factor Analysis Using the EM Algorithm," we show how $p(\mathbf{z})$ can be modeled using a factor analyzer and a mixture of factor analyzers.

3.1 The Transformed Gaussian

To model noisy transformed images of just one appearance, we choose $p(\mathbf{z})$ to be a Gaussian distribution on the pixel intensities:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \mathbf{\Phi}), \quad (3)$$

where $\boldsymbol{\mu}$ is the mean of the Gaussian and $\mathbf{\Phi}$ is the covariance matrix. We usually take $\mathbf{\Phi}$ to be diagonal to reduce the number of parameters that need to be estimated. From (1), the distribution on the observed pixels and the latent pixels given the transformation is

$$p(\mathbf{x}, \mathbf{z}|\mathbf{T}) = \mathcal{N}(\mathbf{x}; \mathbf{T}\mathbf{z}, \mathbf{\Psi})\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \mathbf{\Phi}). \quad (4)$$

The two parameters $\mathbf{\Phi}$ and $\mathbf{\Psi}$ represent two very different types of noise. The noise modeled by $\mathbf{\Phi}$ is added *before* the transformation is applied, whereas the noise modeled by $\mathbf{\Psi}$ is added *after* the transformation is applied. So, whereas $\mathbf{\Phi}$ models pretransformation noise such as background clutter and noisy parts of an object (e.g., blinking eyes), $\mathbf{\Psi}$ models post-transformation noise such as fixed occlusions, window reflections and sensor noise.

Fig. 1b shows hand-crafted parameters of a transformed Gaussian that models a face appearing at different positions in the frame. $\boldsymbol{\mu}$ is shown in raster-scan format. $\mathbf{\Phi}$ is diagonal and the figure shows the diagonal elements of $\mathbf{\Phi}$ in raster-scan format, with large variances painted bright and small variances painted dark. The variance map indicates that the head region is modeled accurately by $\boldsymbol{\mu}$, whereas the surrounding region is not. Fig. 1c shows one configuration of the variables in the model, drawn from the above joint distribution. First, $\mathbf{z}$ is drawn by adding independent Gaussian noise to $\boldsymbol{\mu}$, with variances given by $\mathbf{\Phi}$. Next, a transformation $\mathbf{T}$ is drawn with probability $\pi_{\mathbf{T}}$. Finally, transformation $\mathbf{T}$ is applied to $\mathbf{z}$ and independent Gaussian noise with variances $\mathbf{\Psi}$ (0 in this case) is added to the pixels to produce $\mathbf{x}$.

3.2 The Transformed Mixture of Gaussians (TMG)

In this section, we show how a cluster index variable can be added to the transformed Gaussian model described above, to obtain a transformed *mixture* of Gaussians (TMG) [9]. In contrast to the transformed Gaussian, which models a single cluster, TMG models multiple clusters. The transformed Gaussian is obtained when the number of clusters in the TMG is set to 1.

Cluster c has mean $\boldsymbol{\mu}_c$ and covariance matrix $\mathbf{\Phi}_c$. We usually take $\mathbf{\Phi}_c$ to be diagonal to reduce the number of parameters that need to be estimated. In the companion article, "Transformation-Invariant Factor Analysis Using the EM Algorithm," we describe how a nondiagonal covariance matrix can be modeled using a relatively small number of parameters.

The generative process begins with a random choice for the cluster index c and the transformation $\mathbf{T}$. Next, a latent image $\mathbf{z}$ is obtained by adding zero-mean Gaussian noise to $\boldsymbol{\mu}_c$, with covariance $\mathbf{\Phi}_c$. Transformation $\mathbf{T}$ is applied to $\mathbf{z}$ and zero-mean Gaussian noise with covariance $\mathbf{\Psi}$ is added to obtain the observed image, $\mathbf{x}$.

The distribution over the observed and latent images given the cluster index and transformation is

$$p(\mathbf{x}, \mathbf{z}|\mathbf{T}, c) = p(\mathbf{x}|\mathbf{T}, \mathbf{z})p(\mathbf{z}|c) = \mathcal{N}(\mathbf{x}; \mathbf{T}\mathbf{z}, \mathbf{\Psi})\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_c, \mathbf{\Phi}_c),$$

and the joint distribution is

$$\begin{aligned} p(\mathbf{x}, \mathbf{z}, \mathbf{T}, c) &= p(\mathbf{x}, \mathbf{z}|\mathbf{T}, c)p(\mathbf{T}, c) \\ &= \mathcal{N}(\mathbf{x}; \mathbf{T}\mathbf{z}, \mathbf{\Psi})\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_c, \mathbf{\Phi}_c)\pi_{\mathbf{T},c}, \end{aligned} \tag{5}$$

where the parameter $\pi_{\mathbf{T},c}$ is the probability of cluster c and transformation $\mathbf{T}$.

The cluster-transformation likelihood $p(\mathbf{x}|\mathbf{T}, c)$ for each discrete combination of cluster index and transformation matrix is computed from $p(\mathbf{x}, \mathbf{z}|\mathbf{T}, c)$ by integrating over $\mathbf{z}$:

$$\begin{aligned} p(\mathbf{x}|\mathbf{T}, c) &= \int_{\mathbf{z}} |d\mathbf{z}|\; p(\mathbf{x}, \mathbf{z}|\mathbf{T}, c) \\ &= \int_{\mathbf{z}} |d\mathbf{z}|\; \mathcal{N}(\mathbf{x}|\mathbf{T}\mathbf{z}, \mathbf{\Psi})\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_c, \mathbf{\Phi}_c), \end{aligned}$$

where $|d\mathbf{z}| = \prod_{m=1}^{M} dz_m$ is a differential volume of the space $\mathbb{R}^M$ in which $\mathbf{z}$ lies. The solution to this integral and other useful Gaussian forms are derived in Appendix A. The solution is

$$p(\mathbf{x}|\mathbf{T}, c) = \mathcal{N}\big(\mathbf{x}; \mathbf{T}\boldsymbol{\mu}_c, \mathbf{T}\mathbf{\Phi}_c\mathbf{T}^\top + \mathbf{\Psi}\big), \tag{6}$$

where "$\top$" indicates matrix transpose. Each transformation $\mathbf{T}$ and cluster c has a corresponding mean image $\mathbf{T}\boldsymbol{\mu}_c$ and covariance matrix $\mathbf{T}\mathbf{\Phi}_c\mathbf{T}^\top + \mathbf{\Psi}$. Notice that the latent noise variances, $\mathbf{\Phi}_c$ are transformed in the frame of the observed image.

The probability of the input $p(\mathbf{x})$ is computed from

$$p(\mathbf{x}) = \sum_{c=1}^{C} \sum_{\mathbf{T}\in\mathcal{T}} \mathcal{N}\big(\mathbf{x}; \mathbf{T}\boldsymbol{\mu}_c, \mathbf{T}\mathbf{\Phi}_c\mathbf{T}^\top + \mathbf{\Psi}\big)\pi_{\mathbf{T},c}.$$

Appendix B describes ways to compute the above quantities efficiently.

We now describe how to perform various probabilistic inferences in the TMG. Given the transformation and cluster index, the remaining variables are Gaussian, so the inferences consist of computing conditional means and covariances. See Appendix B for details about how to compute these expectations efficiently.

3.2.1 Inferring the Cluster Index and Transformation

Using the cluster-transformation likelihood and input probability computed as shown above, the posterior probability over each configuration of the cluster index and transformation is computed as follows: $P(\mathbf{T}, c|\mathbf{x}) = p(\mathbf{x}|\mathbf{T}, c)\pi_{\mathbf{T},c}/p(\mathbf{x})$. The most probable cluster index and transformation can be selected to assign the input to a cluster and normalize the input.

From $P(\mathbf{T}, c|\mathbf{x})$, the marginal probabilities of the transformations and cluster indices are easily computed: $P(\mathbf{T}|\mathbf{x}) = \sum_{c=1}^{C} P(\mathbf{T}, c|\mathbf{x})$, $P(c|\mathbf{x}) = \sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}, c|\mathbf{x})$. Also, the class-conditional probability of transformation $\mathbf{T}$ can be computed from $P(\mathbf{T}|c, \mathbf{x}) = P(\mathbf{T}, c|\mathbf{x})/P(c|\mathbf{x})$.

To normalize the input, the maximum a posteriori transformation $\mathbf{T}^* = \mathrm{argmax}_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})$ can be selected and the inverse transformation can be applied to the input: $\mathbf{x}^* = \mathbf{T}^{*-1}\mathbf{x}$. If $\mathbf{T}^{*-1}$ does not exist (e.g., the latent image is larger than the observed image or the transformation erases pixels from the latent image), a pseudoinverse can be used. In fact, as described below, the probability model provides a more principled alternative that does not require the use of a pseudoinverse. Also, the density of the latent image $p(\mathbf{z})$ can be used to regularize the input, i.e., remove some types of noise from the input.

3.2.2 Inferring the Latent Image

The above approach to normalizing the input simply reverses the maximum a posteriori transformation, leaving noise preserved in the normalized image. Since the latent image $\mathbf{z}$ does not contain post-transformation noise such as sensor noise and fixed occlusions, the posterior over $\mathbf{z}$ can be used to normalize the input and remove post-transformation noise.

An efficient way to write the posterior over all hidden variables, $p(\mathbf{z}, \mathbf{T}, c|\mathbf{x})$, is

$$p(\mathbf{z}, \mathbf{T}, c|\mathbf{x}) = P(\mathbf{T}, c|\mathbf{x})p(\mathbf{z}|\mathbf{T}, c, \mathbf{x}). \quad (7)$$

$P(\mathbf{T}, c|\mathbf{x})$ is computed as shown above and since $\mathbf{x}$ and $\mathbf{z}$ are jointly Gaussian under $p(\mathbf{x}, \mathbf{z}|\mathbf{T}, c)$, $\mathbf{z}$ is Gaussian under $p(\mathbf{z}|\mathbf{T}, c, \mathbf{x})$. From the forms derived in Appendix A, we find that the covariance of $\mathbf{z}$ given $\mathbf{T}$, c, and $\mathbf{x}$ is

$$\mathrm{COV}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}] = \left(\mathbf{\Phi}_c^{-1} + \mathbf{T}^\top \mathbf{\Psi}^{-1} \mathbf{T}\right)^{-1}.$$

The inverse covariances add, but the post-transformation inverse covariance $\mathbf{\Psi}^{-1}$ must be appropriately transformed first.

The mean of $\mathbf{z}$ given $\mathbf{T}$, c and $\mathbf{x}$ is

$$\mathrm{E}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}] = \mathrm{COV}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}]\left(\mathbf{\Phi}_c^{-1}\boldsymbol{\mu}_c + \mathbf{T}^\top \mathbf{\Psi}^{-1}\mathbf{x}\right),$$

which is the weighted sum of the cluster center and the transformed input, where the weights are the inverse covariance matrices from above. In regions where the post-transformation noise $\mathbf{\Psi}$ is large relative to the pretransformation noise $\mathbf{\Phi}_c$, the cluster center $\boldsymbol{\mu}_c$ is used to "fill in" the latent image. In regions where the post-transformation noise $\mathbf{\Psi}$ is small relative to the pretransformation noise $\mathbf{\Phi}_c$, the observed data $\mathbf{x}$ determines the latent image.

For cluster c, the mean of the latent image is computed by averaging over the transformations:

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

where $P(\mathbf{T}|c, \mathbf{x})$ and $\mathrm{E}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}]$ are computed as shown above.

Since $p(\mathbf{z}|\mathbf{T}, c, \mathbf{x})$ is Gaussian, the posterior over $\mathbf{z}$ is a mixture of $C \cdot |\mathcal{T}|$ Gaussians. If a point estimate of $\mathbf{z}$ is needed (e.g., to show a picture), obvious alternatives are the mode of the distribution and the expected value of $\mathbf{z}$. Computing the mode of a mixture of Gaussians is not trivial, but a fast approximation is to pick the Gaussian that has the highest posterior probability and output its mean. That is, the transformation $\mathbf{T}^*$ and cluster index c^* that are most probable under $p(\mathbf{T}, c|\mathbf{x})$ are selected and the corresponding mean $\mathrm{E}[\mathbf{z}|\mathbf{T}^*, c^*, \mathbf{x}]$ is used as the point estimate.

However, if $P(\mathbf{T}, c|\mathbf{x})$ is broad (which occurs, e.g., early on in learning when the parameters are not well-determined), the mode is not a good estimate, because it has high variance. The expected value of $\mathbf{z}$ is a better estimate

$$\mathrm{E}[\mathbf{z}|\mathbf{x}] = \sum_{c=1}^{C} P(c|\mathbf{x})\mathrm{E}[\mathbf{z}|c, \mathbf{x}],$$

which is a weighted sum of the posterior mean for each class. This is a normalized version of the input, with post-transformation noise removed.

3.2.3 Inferring the Pretransformation Noise

The pretransformation noise for cluster c accounts for the difference between $\mathbf{z}$ and $\boldsymbol{\mu}_c$. As described above, for an input $\mathbf{x}$, there is uncertainty about the value of $\mathbf{z}$. However, we can compute the mean-squared deviation of the noise, $\mathrm{E}[\mathrm{diag}((\mathbf{z} - \boldsymbol{\mu}_c)(\mathbf{z} - \boldsymbol{\mu}_c)^\top)|c, \mathbf{x}]$. The mean-squared deviation is computed by averaging over the mean-squared deviation for each transformation

$$\mathrm{E}[\mathrm{diag}\Big((\mathbf{z} - \boldsymbol{\mu}_c)(\mathbf{z} - \boldsymbol{\mu}_c)^\top\Big)|c, \mathbf{x}] = \sum_{\mathbf{T} \in \mathcal{T}} P(\mathbf{T}|c, \mathbf{x})\mathrm{E}[\mathrm{diag}\Big((\mathbf{z} - \boldsymbol{\mu}_c)(\mathbf{z} - \boldsymbol{\mu}_c)^\top\Big)|c, \mathbf{T}, \mathbf{x}].$$

The mean-squared deviation for each transformation is given by the squared difference between the mean of $\mathbf{z}$ and $\boldsymbol{\mu}_c$, plus the variance of $\mathbf{z}$

$$\mathrm{E}[\mathrm{diag}\Big((\mathbf{z} - \boldsymbol{\mu}_c)(\mathbf{z} - \boldsymbol{\mu}_c)^\top\Big)|c, \mathbf{T}, \mathbf{x}] = \mathrm{diag}\Big((\mathrm{E}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}] - \boldsymbol{\mu}_c)(\mathrm{E}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}] - \boldsymbol{\mu}_c)^\top\Big) + \mathrm{diag}(\mathrm{COV}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}]).$$

During learning, the mean-squared deviation of the pretransformation noise is used to re-estimate the diagonal pretransformation noise covariance matrix, $\mathbf{\Phi}_c$.

3.2.4 Inferring the Transformed Latent Image

It is very useful to be able to remove post-transformation noise from input data, without also normalizing for transformations. For example, in video, we may want to remove a foreground obstruction that occurs at a fixed position relative to the camera (see Section 4.6). Removal of post-transformation noise is accomplished by averaging the expected value of the transformed latent image for each class: $\mathrm{E}[\mathbf{Tz}|\mathbf{x}] = \sum_{c=1}^{C} P(c|\mathbf{x})\mathrm{E}[\mathbf{Tz}|c, \mathbf{x}]$. $P(c|\mathbf{x})$ is computed as shown above and $\mathrm{E}[\mathbf{Tz}|c, \mathbf{x}] = \sum_{\mathbf{T} \in \mathcal{T}} P(\mathbf{T}|c, \mathbf{x})\mathbf{T}\mathrm{E}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}]$, where $P(\mathbf{T}|c, \mathbf{x})$ and $\mathrm{E}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}]$ are computed as shown above.

3.2.5 Inferring the Post-Transformation Noise

Post-transformation noise accounts for the difference between the transformed latent image, $\mathbf{Tz}$, and the input $\mathbf{x}$. For an input $\mathbf{x}$, there is uncertainty about the values of the cluster index c, the latent image $\mathbf{z}$ and the transformation $\mathbf{T}$. The mean-squared deviation between the observed image and the model's reconstruction of the observed image without post-transformation noise is given by averaging over the class-conditional deviations:

$$\mathrm{E}[\mathrm{diag}\Big((\mathbf{x} - \mathbf{Tz})(\mathbf{x} - \mathbf{Tz})^\top\Big)|\mathbf{x}] = \sum_{c=1}^{C} P(c|\mathbf{x})\mathrm{E}[\mathrm{diag}\Big((\mathbf{x} - \mathbf{Tz})(\mathbf{x} - \mathbf{Tz})^\top\Big)|c, \mathbf{x}].$$

The class-conditional deviation is given by averaging over the transformations:

$$\mathrm{E}[\mathrm{diag}\Big((\mathbf{x} - \mathbf{Tz})(\mathbf{x} - \mathbf{Tz})^\top\Big)|c, \mathbf{x}] = \sum_{\mathbf{T} \in \mathcal{T}} P\Big(\mathbf{T}|c, \mathbf{x})\mathrm{E}[\mathrm{diag}((\mathbf{x} - \mathbf{Tz})(\mathbf{x} - \mathbf{Tz})^\top\Big)|\mathbf{T}, c, \mathbf{x}].$$

It is straightforward to show that the mean-squared deviation for transformation $\mathbf{T}$ and cluster c is

$$\mathrm{E}[\mathrm{diag}\Big((\mathbf{x} - \mathbf{Tz})(\mathbf{x} - \mathbf{Tz})^\top\Big)|\mathbf{T}, c, \mathbf{x}] = \mathrm{diag}\Big((\mathbf{x} - \mathbf{T}\mathrm{E}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}])(\mathbf{x} - \mathbf{T}\mathrm{E}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}])^\top\Big) + \mathrm{diag}\big(\mathbf{T}\mathrm{COV}[\mathbf{z}|\mathbf{T}, c, \mathbf{x}]\mathbf{T}^\top\big).$$

The mean-squared deviation of the post-transformation noise can be used to identify regions of the image that are accounted for by post-transformation noise. Also, during

learning, it is used to re-estimate the diagonal post-transformation noise covariance matrix $\mathbf{\Psi}$.

3.3 Learning Using the EM Algorithm

We now present an EM algorithm [5] for the transformed mixture of Gaussians (TMG) that starts with randomly initialized parameters and then performs maximum-likelihood parameter estimation. (Here, "likelihood" refers to the parameters, not the hidden variables as in the previous section.) EM for the transformed Gaussian is obtained simply by setting the number of clusters in the TMG to 1.

The only input to the algorithm is the training set $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(J)}$, the set of possible transformations $\mathcal{T}$, and the number of clusters C. MATLAB software is available at http://www.psi.toronto.edu.

Experiments indicate that the algorithm does *not* require careful initialization of the parameters and we usually initialize the parameters to random values. However, as with a standard mixture of Gaussians, it is a good idea to initialize the parameters taking into account the scale of the data. For example, if the pixel intensities in the training set have mean μ and variance σ^2, the elements of $\boldsymbol{\mu}_c$ can be set to μ plus Gaussian noise with variance σ^2, and the diagonal elements of $\mathbf{\Phi}$ and $\mathbf{\Psi}$ can be set to $5\sigma^2$, to ensure the data has a reasonable likelihood initially. We usually set the mixing proportions $\pi_{\mathbf{T},c}$ to be uniform.

After initialization, the EM algorithm proceeds by alternating between an E-step and an M-step for a fixed number of iterations (e.g., 30), or until convergence is detected automatically by monitoring the probability of the data.

In the E-step, the model parameters are assumed to be correct, and for each input image, probabilistic inference is used to fill in the values of the unobserved variables—the latent image $\mathbf{z}$, the image cluster index c, and the transformation $\mathbf{T}$. While passing through the training set, the algorithm accumulates sufficient statistics for computing the parameters that maximize the joint probability of the observed variables and the filled-in variables. In the M-step, the sufficient statistics accumulated in the E-step are used to compute a new set of parameters.

In fact, for each input image, the E-step fills in the unobserved variables with a *distribution* over plausible configurations (the posterior distribution), not just individual configurations. This is an important aspect of the EM algorithm. Initially, the parameters are a very poor representation of the data (we initialized them randomly) so, any single configuration of the unobserved variables (e.g., the most probable configuration under the posterior) will very likely be the wrong configuration.

Readers who are familiar with the EM algorithm for a standard mixture of Gaussians should keep in mind that the EM algorithm for the TMG is similar to a constrained, reparameterized version of the EM algorithm for a standard mixture of Gaussians. A notable difference is that the noise is separated into post-transformation noise $\mathbf{\Psi}$ and pretransformation noise $\mathbf{\Phi}$.

The hidden variables in the TMG are c, $\mathbf{T}$, and $\mathbf{z}$. Assuming the distribution used for filling in the hidden variables for training case $\mathbf{x}^{(j)}$ is $q(c, \mathbf{T}, \mathbf{z}|\mathbf{x}^{(j)})$, the system of equations for the parameter updates are obtained from

$$\sum_{j=1}^{J} \sum_{c=1}^{C} \sum_{\mathbf{T}\in\mathcal{T}} \int_{\mathbf{z}} |d\mathbf{z}|\; q\Big(c, \mathbf{T}, \mathbf{z}|\mathbf{x}^{(j)}\Big) \frac{\partial}{\partial\theta} \ln p\Big(\mathbf{x}^{(j)}, \mathbf{z}, \mathbf{T}, c\Big) = 0,$$

where θ is one of $\pi_{\mathbf{T},c}$, $\boldsymbol{\mu}_c$, $\mathbf{\Phi}_c$, and $\mathbf{\Psi}$, and there is one equation for each parameter.

Each equation is solved by taking the appropriate derivative using the formula for $p(\mathbf{x}^{(j)}, \mathbf{z}, \mathbf{T}, c)$ given in (5) and then numerically averaging over the hidden variables with respect to $q(\cdot)$. Exact EM is obtained by setting $q(c, \mathbf{T}, \mathbf{z}|\mathbf{x}^{(j)}) = p(c, \mathbf{T}, \mathbf{z}|\mathbf{x}^{(j)})$, where $p(c, \mathbf{T}, \mathbf{z}|\mathbf{x})$ is the posterior distribution (7) computed using the model parameters from the previous iteration.

Solving for $\pi_{\mathbf{T},c}$ (using a Lagrange multiplier to ensure that $\sum_{c=1}^{C} \sum_{\mathbf{T}\in\mathcal{T}} \pi_{\mathbf{T},c} = 1$), we obtain the parameter update

$$\pi_{\mathbf{T},c} \leftarrow \frac{1}{J} \sum_{j=1}^{J} P\Big(\mathbf{T}, c|\mathbf{x}^{(j)}\Big),$$

where $P(\mathbf{T}, c|\mathbf{x})$ is computed as shown in Section 3.2.1. So, the prior probability of transformation $\mathbf{T}$ and cluster c is set equal to the average of the posterior probability.

Usually, there is not enough data to accurately estimate the distribution over transformations. In this case, the distribution over transformations can be assumed to be uniform and the above update is replaced by

$$\pi_{\mathbf{T},c} \leftarrow \frac{1}{|\mathcal{T}|} \frac{1}{J} \sum_{j=1}^{J} P\Big(c|\mathbf{x}^{(j)}\Big),$$

where $|\mathcal{T}|$ is the number of transformations, and $P(c|\mathbf{x})$ is computed as shown in Section 3.2.1. Alternatively, $\pi_{\mathbf{T},c}$ can be regularized using prior knowledge about which transformations should have similar probabilities.

The update for the mean of cluster c is given by the average value of the posterior mean of the latent image (the normalized version of the input, with post-transformation noise removed)

$$\boldsymbol{\mu}_c \leftarrow \frac{1}{J} \sum_{j=1}^{J} \mathrm{E}\Big[\mathbf{z}|c, \mathbf{x}^{(j)}\Big].$$

The posterior mean of the latent image for cluster c, $\mathrm{E}[\mathbf{z}|c, \mathbf{x}]$, is computed as shown in Section 3.2.2.

The update for the covariance of the pretransformation noise for cluster c is given by the average of the mean-squared deviation between the latent image and the cluster center

$$\mathbf{\Phi}_c \leftarrow \frac{1}{J} \sum_{j=1}^{J} \mathrm{E}\Big[\mathrm{diag}\Big((\mathbf{z} - \boldsymbol{\mu}_c)(\mathbf{z} - \boldsymbol{\mu}_c)^{\top}\Big)|c, \mathbf{x}^{(j)}\Big].$$

The mean-squared deviation $\mathrm{E}[\mathrm{diag}((\mathbf{z} - \boldsymbol{\mu}_c)(\mathbf{z} - \boldsymbol{\mu}_c)^{\top})|c, \mathbf{x}]$ is computed as shown in Section 3.2.3. In the above expression, the value of $\boldsymbol{\mu}_c$ from the previous iteration can be used or the most recently updated value of $\boldsymbol{\mu}_c$ can be used.

The update for the covariance of the post-transformation noise is given by the average of the mean-squared deviation between the input and the transformed latent image

$$\mathbf{\Psi} \leftarrow \frac{1}{J} \sum_{j=1}^{J} \sum_{c=1}^{C} \mathrm{E}\Big[\mathrm{diag}\Big((\mathbf{x}^{(j)} - \mathbf{T}\mathbf{z})(\mathbf{x}^{(j)} - \mathbf{T}\mathbf{z})^{\top}\Big)|c, \mathbf{x}^{(j)}\Big].$$

The mean-squared deviation $\mathrm{E}[\mathrm{diag}((\mathbf{x}-\mathbf{Tz})(\mathbf{x}-\mathbf{Tz})^\top)|c,\mathbf{x}]$ is computed as shown in Section 3.2.5.

In order to avoid overfitting the noise variances, it is often useful to set the diagonal elements of $\mathbf{\Phi}_c$ and $\mathbf{\Psi}$ that are below some ϵ equal to ϵ. For data with pixel intensities between 0 and 1, we set $\epsilon = 0.0001$.

4 Experimental Results on TMG

We report results that demonstrate the usefulness of the transformed mixture of Gaussians (TMG) and provide evidence that TMG overcomes deficiencies in standard mixture modeling when the data is transformed. In particular, we find that for data sets that contain transformations of a known type, TMG does not require careful selection of initial training conditions, TMG is faster and produces better density models than standard mixture modeling, TMG is faster and produces better density models even when standard mixture modeling is provided with more training data, and TMG is able to identify pretransformation noise and post-transformation noise, whereas standard mixture models do not distinguish these noise sources. To focus on these issues, we restrict this section to problems where the data is a set of images and the set of transformations is all possible translations in the image (with wrap-around).

In all cases (unless otherwise noted), before training, the image means $\boldsymbol{\mu}$ are initialized to the mean of the data plus Gaussian noise with standard deviation 0.1 (the pixel intensities are in the range $[0,1)$). The variances are initialized to the value 5. The class probabilities are initialized to be uniform and the transformation probabilities are constrained to be equal and independent of the class.

4.1 Filtering Images from a Scanning Electron Microscope

Due to high variance in electron emission rate and modulation of this variance by the imaged material [14], images produced by scanning electron microscopes can be highly distorted. Fig. 2a shows some examples from a set of 230, 140×56 images recorded in sequence from an electron microscope.

To learn a normalized appearance of the sample, we trained a transformation-invariant Gaussian (1-cluster TMG) on the 230 images using 50 iterations of EM. This took 10 minutes on a 1GHz PIII computer. The set of transformations consists of all $140 \cdot 56$ translations, with wrap-around. In fact, this set of transformations does not perfectly match the physical process, since each microscope image is formed by a linear combination of deformed images. Also, the deformations may be slightly nonuniform.

Fig. 2b shows the means found by each of four runs of EM, where the runs differ only in the seed used to randomly initialize the parameters. To make these means easier to view, after learning, each mean image was translated by the most probable transformation in the training data. Clearly, the algorithm is not very sensitive to initial conditions. The average of the means from the four runs is shown in Fig. 2c, and it is only slightly clearer than any one of the means.

We compare the result of TMG with two other approaches: straight averaging and averaging after greedy alignment. Averaging the images in a straightforward fashion, produces the mean image shown in Fig. 2d. The image produced by TMG (Fig. 2c) shows several details that are obfuscated in the straight average (Fig. 2d). The number of parameters in the TMG is *equal* to the number of parameters in the straight average, plus 1. So, it is very likely that these details are present in the true sample and are not due to overfitting.

We also compare the mean of TMG with the average image obtained after each image in the set is aligned to the first image by maximizing correlation. Fig. 2e shows the average image obtained in this fashion. Although this image is clearer than the straight average, TMG produces a mean that is significantly sharper. We tried several versions of the greedy alignment method, including methods based on Euclidean distance, mean-normalized correlation, variance-normalized correlation, and combined mean and variance-normalized correlation. All greedy methods produced images similar to the one shown in Fig. 2e.

4.2 Quantitative Comparison of TMG versus Mixtures of Gaussians

Often, interpreting the model parameters is not too important but we want to train density models for anomaly detection and pattern classification. If $P(i)$ is the prior probability of class i and $p(\mathbf{x}|i)$ is a density model of the input $\mathbf{x}$ for class i, a new input can be classified by picking the most probable class in the posterior, $P(i|\mathbf{x}) = \big(P(i)p(\mathbf{x}|i)\big)/\big(\sum_{i'} P(i')p(\mathbf{x}|i')\big)$. In this case, our main interest is in obtaining good density models.

Here, we compare the classification error rate obtained using a single transformed Gaussian for each class with the error rate obtained using a standard mixture model for each class. To address the question of whether increasing the number of Gaussians in a standard mixture model can make it competitive with a single transformed Gaussian, we examine mixture models with varying numbers of Gaussians. Generally, more training data leads to better density models, so we plot the error rates as a function of the number of training cases.

Our goal here is to determine whether including transformation as a hidden variable performs better than using a large mixture model. So, we use synthetic data, where, within each class, the only sources of variability are translations and noise. It turns out that the standard mixture model requires a large number of training cases to be competitive (in error rate) with the transformed Gaussian. This is another reason we use synthetic data, since we are able to generate as many training examples as needed.

Fig. 3a shows some examples of the data. Within each of the four classes, the pattern appears at one of 25 positions. Separately from the training data, we generated 1,000 test cases.

For training set sizes ranging from four cases (one from each class) to 2,000 cases, we generated training data, trained each of the four transformed Gaussian models using 50 iterations of EM, and measured the test error rate as a function of the number of training cases. For small training sets, the selection of a training set introduces variability in the error rate, so we repeated the above procedure five times and computed the average of the five curves.

The above training and testing procedure was repeated for Gaussian mixture models with 1, 2, 5, 10, 20, 30, and 40 Gaussians. Since each pattern can occur in one of 25 positions, a 25-Gaussian mixture model can obtain the minimum Bayes error rate. We include results on models with up to 40 Gaussians to allow for local minima.

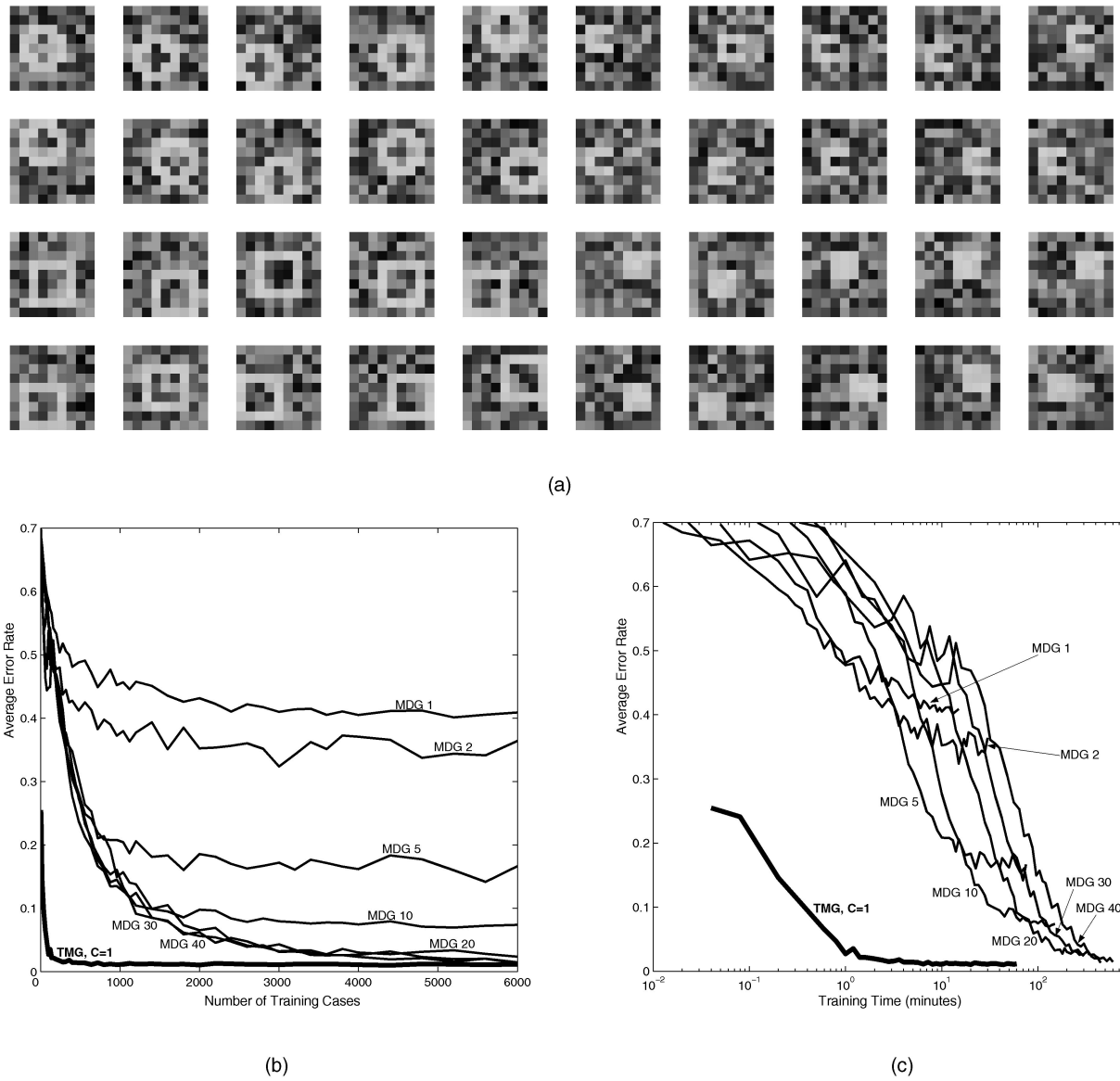


Fig. 3. (a) Examples of 9×9 images from each of four classes of synthetic data. (b) Average error rate versus number of training cases when Bayes rule is used to classify patterns using class-conditional density models, including TMG with one cluster (labeled "TMG"), and standard Gaussian mixture models (labeled "MDG") with 1, 2, 5, 10, 20, 30, and 40 Gaussians. (c) Average error rate versus time (log scale) required to train the density models.

Fig. 3b shows the average error rate as a function of the number of training cases for the transformed Gaussian model and the mixture models of various sizes. For a given amount of training data, the transformed Gaussian clearly performs better than all sizes of mixture model. The mixture model classifiers require large amounts of data to properly determine which parts of the image are structure and which parts of the image are noise. In contrast, the transformed Gaussian is able to align the patterns and determine the highly noisy regions of the image.

4.3 Why Not Train a Mixture of Gaussians on Transformed Data?

When training data is limited, one way to obtain a density model that allows for transformations in the input is to apply transformations to each of the original training cases, producing a new, extended training set. A standard density model (e.g., a mixture of Gaussians) can then be estimated from the extended training set.

For 9×9 images like the ones shown in Fig. 3a, each original training case is translated to each of 81 positions, with wrap-around, producing a new training set that is 81 times larger than the original training set. If wrap-around is not used, care must be taken to appropriately pad portions of the image that are not accounted for by the original image.

While transforming the training data is an appealingly simple approach, it suffers from three significant deficiencies. First, the techniques described in Appendix B can be used to make the EM algorithm in the TMG run *faster* than standard mixture modeling on an extended training set. Second, post-transformation noise (noise that is in a fixed position relative to the camera) gets transformed along with the object when creating the extended training set. So, post-transformation noise cannot be distinguished from pretransformation noise (whose position is transformed with the object), worsening

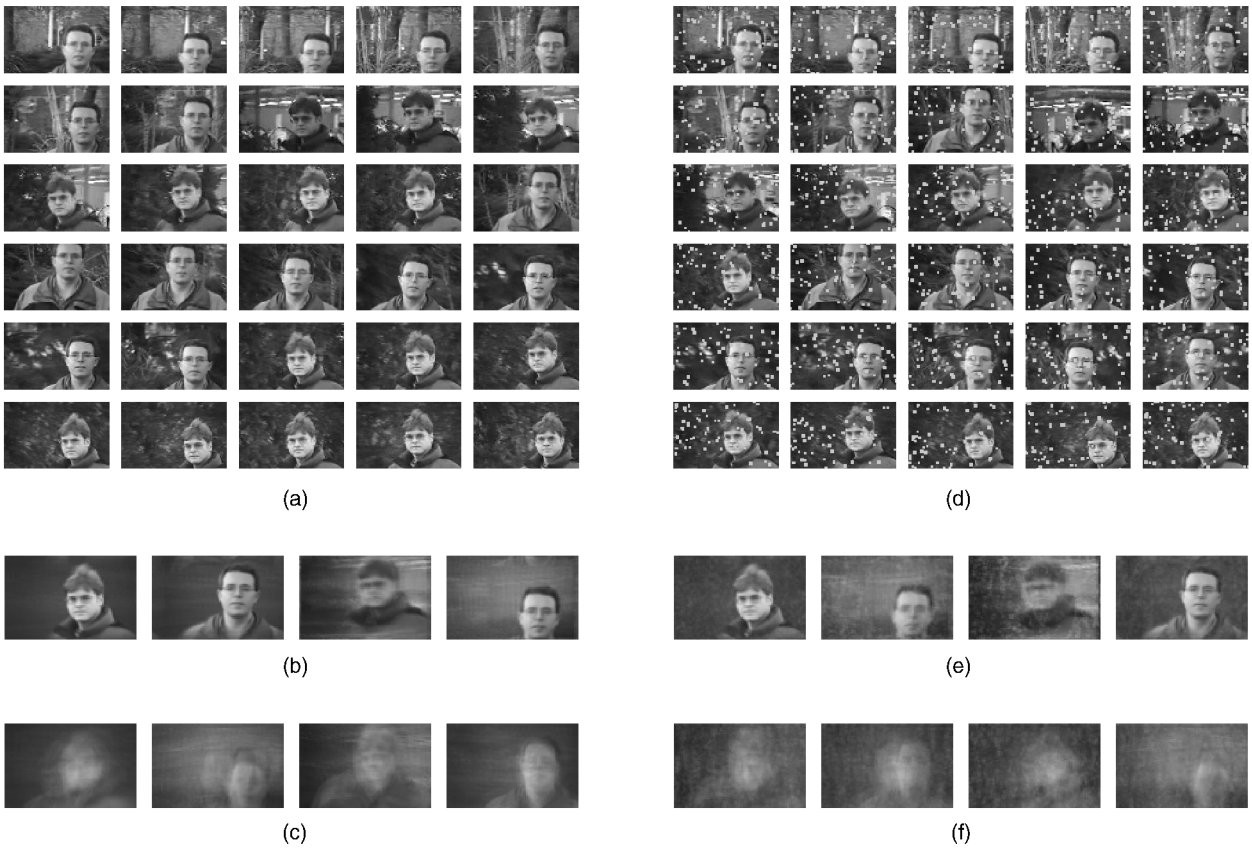


Fig. 4. (a) Images of size 88×56 pixels from a 200-frame video that cuts back and forth between each of two people walking across a cluttered background. (b) Cluster means after 20 iterations of EM in a 4-cluster TMG where the set of transformations is all 4,928 translations with wrap-around. (Execution time: five minutes on a 1 GHz PIII.) Each mean image is automatically centered for easy viewing, by translating the image by the most probable transformation across the data. (c) Cluster means after 20 iterations of EM in a standard mixture of four Gaussians. (d) The video sequence was modified to include artificial snow falling in front of the scene. (e) Cluster means found by TMG. (f) Cluster means found by a standard mixture of Gaussians.

the quality of the density model (see Section 4.6). Third, training a large mixture model on the extended data set will produce cluster centers of the same pattern at different positions. Higher-level processing (by human or by machine) is much easier if the centers are grouped into equivalence classes. Whereas TMG does this automatically, standard mixture modeling on an extended training set does not.

We demonstrate the first deficiency of standard mixture modeling from above, using the same experimental set-up as in the previous section. In fact, we allow the standard mixture model to "cheat": Instead of simply translating each original training case to 81 different positions, we extend the training set to 81 times its original size by generating new images. This procedure makes learning easier for the standard mixture model, because instead of translating the noise along with the pattern 81 times, new noise is generated, so the model can more easily distinguish noisy regions from patterned regions.

As above, we try various sizes of training set and evaluate the average classification error rate of the different models. Fig. 4c shows the average error rate as a function of the time needed to train the four density models used for classification. For the same error rate, TMG is 2 to 3 *orders of magnitude* faster than standard mixture modeling.

4.4 Clustering Images of Faces in Severe Noise

Fig. 4a shows images of size 88×56 pixels from a 200-frame video that cuts back and forth between each of two people walking across a cluttered background. We report results for a 4-cluster TMG, where the set of transformations is all 4,928 possible translations with wrap-around.

After 20 iterations of EM (which took five minutes on a 1 GHz PIII machine running MATLAB) the cluster means clearly show the two subjects and suppress the background clutter, as shown in Fig. 4b. Each mean image is automatically centered for easy viewing, by translating the image by the most probable transformation across the data. Two versions of each person are found: for one person, two slightly different lighting conditions are found; for the other person, a version with the jacket and a version without the jacket are found.

Fig. 4c shows the means after 20 iterations of EM in a standard mixture of four Gaussians. These means are significantly blurrier than the means found by the TMG.

Even when the images are obfuscated by severe snow, TMG is able to learn means that are quite clear. Figs. 4d, 4e, and 4f show the results for the exact same experimental setup as above, except for a modification of the training data, as shown in Fig. 4d.

4.5 Experimental Evidence of Robustness

In the above problem, clustering is made easier by the fact that the two patterns (faces) look quite different. In this section, we study the problem of clustering different facial *poses* of the same person, and examine the robustness of the algorithm to initial conditions and the number of clusters. Fig. 5a shows

examples from an unordered training set of 400 images containing the same person, but with different poses. Here, there is more similarity in appearance between the different classes, so separating the classes is more difficult.

To show the EM algorithm in the TMG model is not too sensitive to initial conditions, we report results on clustering the images using eight different sizes of TMG and, in each case, four different random initializations of the model parameters.

In all training runs, we used 30 iterations of EM. The pixel intensities are in the range $[0, 1)$, so to ensure that initially the data has reasonable likelihood, we initialized all noise variances to 5 (anything over 1 would have produced similar results). We initialized the means of the latent image to the mean of the training data, plus Gaussian noise with standard deviation 0.1. Initializing the means in this way ensures they are in the vicinity of the data. The class probabilities were initialized to uniform values and the transformation probabilities were set to uniform values and not learned.

Fig. 5b shows the means after learning. They have been translated by the most probable transformation in the data, which tends to center the faces, making them easier to examine. We observe that TMG finds very similar solutions for the same size of model. We even observe the solutions are similar from one model size to the next. Comparing one model size to the next, it is possible to see that particular blurry cluster centers are refined into two more detailed cluster centers.

For this data, we see that there appears to be two types of solution—one that accounts for the bright rectangle-shaped blob and one that does not. In fact, by examining the learned mixing proportions, it is possible to detect that the prototype with the rectangle is unlikely since its mixing proportion is always significantly lower than the mixing proportions for the other prototypes.

4.6 Learning to Separate Pretransformation Noise from Post-Transformation Noise

We give results that demonstrate the importance of having two very different noise models for pretransformation noise and post-transformation noise.

Fig. 6a shows a training set of 30, 88×56 images. The data contain an artificial obstruction at a fixed position relative to the camera, placed in front of images from a video of a person walking across a cluttered background.

After training a TMG, probabilistic inference can be used to remove the fixed obstruction from the input images, as described in Section 3.2.4. For each training case shown in Fig. 6a, $\mathrm{E}[\mathbf{T}\mathbf{z}|\mathbf{x}]$ is computed and shown in Fig. 6b. The obstruction has been successfully removed in most cases.

The reason that TMG can remove the fixed obstruction while preserving the background clutter is that TMG accounts for pretransformation noise and post-transformation noise separately. To clarify this issue, we examine the model parameters found by straight averaging, TMG with *one* noise model, and the TMG model used to obtain the results in Fig. 6b.

Fig. 6c shows the pixel means and variances of the training images as intensity maps. Obviously, the mean of the data includes the obstruction since the mean of the data does not account for translations.

Fig. 6d shows the mean and variance map found by a 1-cluster TMG that does not distinguish between pre- and post-transformation noise. In this TMG, we set $\mathbf{\Psi} = 0$, which forces the model to use the pretransformation noise model $\mathbf{\Phi}$ to model both pre and post-transformation noise. Since the TMG cannot account for the obstruction as post-transformation noise, the model learns the appearance of the obstruction.

For this data, pretransformation noise is needed to model the background noise, since the allowed locations of background noise translate with the position of the head. Post-transformation noise is needed to account for the fixed obstruction since the position of the obstruction is fixed relative to the frame.

Fig. 6e shows the mean μ, pretransformation variance $\mathbf{\Phi}$ and post-transformation variance $\mathbf{\Psi}$ after 30 iterations of EM in a 1-cluster TMG, where $\mathbf{\Psi}$ is not forced to be 0. The post-transformation noise accounts for the obstruction, allowing the mean to properly model the details of the face.

5 Summary

In many learning applications, we know beforehand that the data includes transformations of an easily specified nature. If a density model is learned directly from the data, the model must account for both the transformations in the data and the more interesting and potentially useful structure.

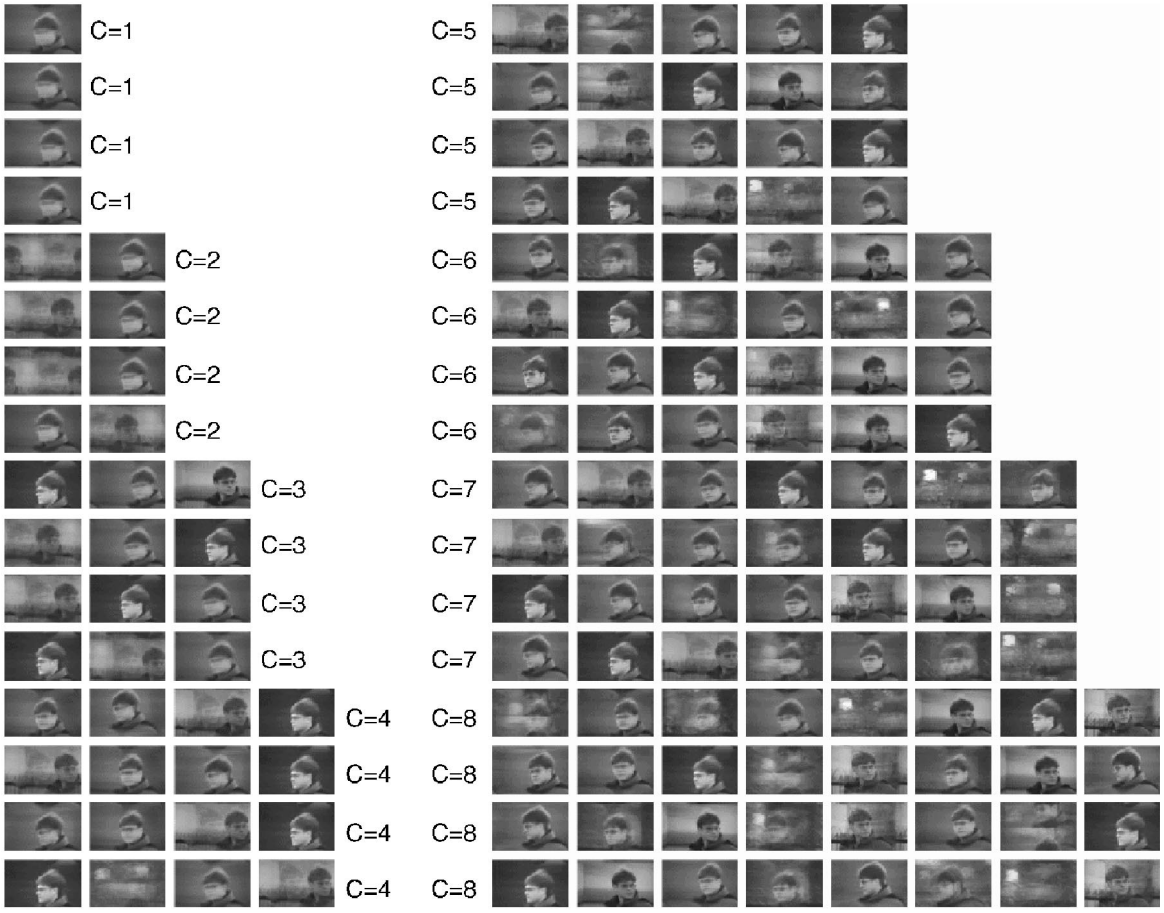
We introduce a way to make standard density models for clustering invariant to local and global transformations in the input. Global transformations are generally nonlinear, so we approximate the manifold of transformed data using a discrete set of points. The resulting latent variable model (the transformed mixture of Gaussians—TMG) contains continuous and discrete variables. However, given the discrete variables, the distribution over the continuous variables is jointly Gaussian, so inference and estimation (via the expectation maximization algorithm) can be performed efficiently.

The algorithm is able to jointly normalize input data for transformations (e.g., translation, shearing, rotation, and scale in images) and cluster the normalized data.

Although discretizing the manifold of transformed data is only an approximation, the approximation avoids the problem of how to perform inference and learning with continuous variables that combine nonlinearly. Approximate inference of continuous variables can be approached in different ways, including Monte Carlo techniques (c.f. [18]) and variational techniques (c.f. [7]). Tenenbaum and Freeman [24] examine models, called "bilinear models," where each hidden variable is a linear function of the data, given the other hidden variables. The authors derive an inference algorithm that iterates between the hidden variables. However, global transformations are generally *very* nonlinear, so the above approximations tend to be either slow or too inexact. Although discretizing the latent variable space seems crude in light of these more sophisticated approximations, the resulting model is simple and inference is surprisingly fast.

In contrast to methods that explain how one observed image differs from another observed image (c.f. [3]), our algorithms explain how the observed image differs from a model of the normalized image. This allows our techniques to properly warp two images to the model, even if the two images are warped versions of nonoverlapping subimages of the model. For example, one observed image could be the left half of a smile, while another observed image could be the right half of a smile. Even though these two images are not warped versions of each other, our algorithms can warp them to the latent density model of the normalize image.

(a)



(b)

Fig. 5. (a) Examples of 88×56 images from a 400-frame video sequence, showing one person with different poses in front of a cluttered background. (b) To show that the TMG is quite insensitive to initialization and behaves well as the number of clusters is increased, we trained 32 TMG models on the set of images. Each group of pictures shows the means found after 30 iterations of EM. The number of clusters in the TMG models range from $C = 1$ to $C = 8$ and for each size of model, the parameters were randomly initialized using four different seeds. Each mean image is automatically centered for easy viewing, by translating the image by the most probable transformation across the data.

Often, data is best described as the result of multiple, interacting causes, as opposed to a single, global cause, such as translation. The most principled way to model multiple causes is to derive approximate inference and learning algorithms in complex probability models [17], [15], [4], [13], [8], [1], [6], [2], [27]. While this approach holds potential for solving complex problems, in many cases, what is needed is an efficient way to remove transformations of a known type from the data. This is what TMG does.

We report extensive experimental results and find that for data sets that contain transformations of a known type, TMG does not require careful selection of initial training conditions; TMG is faster and produces better density models than standard mixture modeling; TMG is faster and produces better density models even when standard mixture modeling is provided with more training data; and TMG is able to identify pretransformation noise and post-transformation noise, whereas standard mixture models do not distinguish these noise sources.

The algorithms can be applied in a variety of domains (e.g., images, audio signals), but we illustrate the algorithms on a variety of difficult tasks in computer vision. For

Fig. 6. (a) An artificial obstruction at a fixed position relative to the camera is placed in front of 30 images from a video of a person walking across a cluttered background. The region containing background noise shifts relative to the camera (depending on the position of the head), whereas the obstruction occurs at at fixed position relative to the camera. (b) After the parameters of the 1-cluster TMG are estimated, probabilistic inference is used to compute $\mathrm{E}[\mathbf{T}\mathbf{z}|\mathbf{x}]$ for each input $\mathbf{x}$, as described in Section 3.2.4. The resulting images have post-transformation noise removed (compare with the images in (a)). (c) The mean image (first picture) and variance map (second picture) estimated directly from the training set. White indicates high variance, black indicates a variance of 0. (d) The mean $\boldsymbol{\mu}$ (first picture) and pretransformation noise variance $\boldsymbol{\Phi}$ (second picture) after 30 iterations of EM in a 1-cluster TMG, where we fix $\boldsymbol{\Psi} = 0$. $\boldsymbol{\Phi}$ models both the background clutter and the fixed distraction. (e) The mean $\boldsymbol{\mu}$ (first picture), pretransformation noise variance $\boldsymbol{\Phi}$ (second picture) and post-transformation noise variance $\boldsymbol{\Psi}$ (third picture) after 30 iterations of EM in a 1-cluster TMG. $\boldsymbol{\Phi}$ models the background clutter, whereas $\boldsymbol{\Psi}$ models the fixed distraction.

example, the transformation-invariant mixture of Gaussians is able to learn different facial poses from a set of outdoor images showing a person walking across a cluttered background with varying lighting conditions. We focus on translational transformations in this paper, but other types of transformation can be used, such as rotation, scale, out-of-plane rotation, and warping in images.

By measuring the data on an appropriate coordinate system, many types of transformation can be represented as shifts in the coordinate system. In these cases, the FFT can be used to tremendously speed up inference and estimation. For example, our current 1GHZ PIII MATLAB implementation of TMG can perform the E step and M step for two clusters and all possible integer-pixel x-y translations in a 320×240 image, at *15 frames per second*.

In the case of time-series data, the transformations at the neighboring time steps influence which transformations are likely in the current time step. In [19], [12], we show how the techniques presented here can be extended to time series.

The number of computations needed for exact inference scales exponentially with the dimensionality of the transformation manifold. If there are n_1 transformations of the first type, n_2 transformations of the second type, etc., exact inference and learning takes order $\prod_i n_i$ time. In [11], we show how a variational technique can be used to decouple the inference of each type of transformation,

producing an inference and learning method that takes order $\sum_i n_i n_{i+1}$ time.

MATLAB scripts for transformation-invariant clustering and component analysis are available on our Web page at http://www.psi.toronto.edu.

In the companion article, "Transformation-Invariant Factor Analysis Using the EM Algorithm," we show how factor analysis (generative modeling akin to PCA) can be performed in a way that is invariant to transformations in the input.

We believe the algorithms presented here will prove to be generally useful for applications that require transformation-invariant clustering.

APPENDIX A

LINEAR ALGEBRA FOR GAUSSIAN FORMS

The following formulas are used repeatedly throughout the paper. Suppose the distribution over two jointly Gaussian vector variables $\mathbf{x}$ and $\mathbf{z}$ is given in the form, $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, where

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{G}\mathbf{z}, \boldsymbol{\Psi}), \quad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Phi}). \tag{8}$$

$\boldsymbol{\mu}$ and $\boldsymbol{\Phi}$ are the mean and covariance of $\mathbf{z}$. $\mathbf{G}$ is a matrix that determines the mean of $\mathbf{x}$ given $\mathbf{z}$. $\boldsymbol{\Psi}$ is the covariance of $\mathbf{x}$ given $\mathbf{z}$. Since $\mathbf{x}$ and $\mathbf{z}$ are jointly Gaussian, all conditional and marginal distributions are Gaussian.

A.1 The Conditional $p(\mathbf{z}|\mathbf{x})$

Since $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{x}, \mathbf{z})/p(\mathbf{x})$ and $p(\mathbf{x})$ is not a function of $\mathbf{z}$, the mean and covariance of the conditional distribution $p(\mathbf{z}|\mathbf{x})$ can be determined by rewriting the exponent of $p(\mathbf{x}, \mathbf{z})$ as a completed square in $\mathbf{z}$. Leaving out a factor of $-1/2$, the exponent is

$$\begin{aligned}(\mathbf{x} - \mathbf{G}\mathbf{z})^\top \boldsymbol{\Psi}^{-1}(\mathbf{x} - \mathbf{G}\mathbf{z}) + (\mathbf{z} - \boldsymbol{\mu})^\top \boldsymbol{\Phi}^{-1}(\mathbf{z} - \boldsymbol{\mu}) \\ = \mathbf{x}^\top \boldsymbol{\Psi}^{-1}\mathbf{x} - 2\mathbf{x}^\top \boldsymbol{\Psi}^{-1}\mathbf{G}\mathbf{z} + \mathbf{z}^\top \mathbf{G}^\top \boldsymbol{\Psi}^{-1}\mathbf{G}\mathbf{z} \\ + \mathbf{z}^\top \boldsymbol{\Phi}^{-1}\mathbf{z} - 2\boldsymbol{\mu}^\top \boldsymbol{\Phi}^{-1}\mathbf{z} + \boldsymbol{\mu}^\top \boldsymbol{\Phi}^{-1}\boldsymbol{\mu} \\ = \mathbf{x}^\top \boldsymbol{\Psi}^{-1}\mathbf{x} + \mathbf{z}^\top \left(\mathbf{G}^\top \boldsymbol{\Psi}^{-1}\mathbf{G} + \boldsymbol{\Phi}^{-1}\right) \\ \mathbf{z} - 2\left(\mathbf{x}^\top \boldsymbol{\Psi}^{-1}\mathbf{G} + \boldsymbol{\mu}^\top \boldsymbol{\Phi}^{-1}\right)\mathbf{z} + \boldsymbol{\mu}^\top \boldsymbol{\Phi}^{-1}\boldsymbol{\mu}.\end{aligned} \tag{9}$$

Defining the symmetric matrix and vector,

$$\boldsymbol{\Omega} = \left(\mathbf{G}^\top \boldsymbol{\Psi}^{-1}\mathbf{G} + \boldsymbol{\Phi}^{-1}\right)^{-1}, \quad \boldsymbol{\eta} = \boldsymbol{\Omega}\left(\mathbf{G}^\top \boldsymbol{\Psi}^{-1}\mathbf{x} + \boldsymbol{\Phi}^{-1}\boldsymbol{\mu}\right), \tag{10}$$

and substituting these into the exponent, we can rewrite the exponent as the following completed square in $\mathbf{z}$: $\mathbf{x}^\top \boldsymbol{\Psi}^{-1}\mathbf{x} + (\mathbf{z} - \boldsymbol{\eta})^\top \boldsymbol{\Omega}^{-1}(\mathbf{z} - \boldsymbol{\eta}) - \boldsymbol{\eta}^\top \boldsymbol{\Omega}^{-1}\boldsymbol{\eta} + \boldsymbol{\mu}^\top \boldsymbol{\Phi}^{-1}\boldsymbol{\mu}$. From this expression, we see that $\boldsymbol{\eta}$ and $\boldsymbol{\Omega}$ are the mean and covariance of $\mathbf{z}$ given $\mathbf{x}$. From (10), the mean of $\mathbf{z}$ given $\mathbf{x}$ is a weighted sum of the mean of $\mathbf{z}$ under the "prior" $p(\mathbf{z})$ and the mean of $\mathbf{z}$ as given by the "likelihood" $p(\mathbf{x}|\mathbf{z})$.

A.2 The Marginal $p(\mathbf{x})$

The mean and covariance of the marginal for $\mathbf{x}$ are most readily determined by simplifying expectations of $\mathbf{x}$. From $p(\mathbf{x}|\mathbf{z})$ given above, $\mathbf{x} = \mathbf{G}\mathbf{z} + \boldsymbol{\nu}$, where $\boldsymbol{\nu}$ is a Gaussian random variable with mean zero and covariance $\boldsymbol{\Psi}$, and $\mathbf{z}$ is a Gaussian random variable with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Phi}$. Using $\mathrm{E}[\cdot]$ to denote an expectation, the mean of $\mathbf{x}$ is $\mathrm{E}[\mathbf{x}] = \mathrm{E}[\mathbf{G}\mathbf{z} + \boldsymbol{\nu}] = \mathbf{G}\mathrm{E}[\mathbf{z}] + \mathrm{E}[\boldsymbol{\nu}] = \mathbf{G}\boldsymbol{\mu}$. The covariance of $\mathbf{x}$, $\mathrm{E}[(\mathbf{x} - \mathbf{G}\boldsymbol{\mu})(\mathbf{x} - \mathbf{G}\boldsymbol{\mu})^\top]$, simplifies to

$$\mathbf{G}\mathrm{E}\left[(\mathbf{z} - \boldsymbol{\mu})(\mathbf{z} - \boldsymbol{\mu})^\top\right]\mathbf{G}^\top + 2\mathrm{E}\left[\boldsymbol{\nu}(\mathbf{z} - \boldsymbol{\mu})^\top\right]\mathbf{G}^\top + \mathrm{E}\left[\boldsymbol{\nu}\boldsymbol{\nu}^\top\right].$$

Since $\mathbf{z}$ and $\boldsymbol{\nu}$ are independent,

$$\mathrm{E}\left[\boldsymbol{\nu}(\mathbf{z} - \boldsymbol{\mu})^\top\right] = \mathrm{E}[\boldsymbol{\nu}]\mathrm{E}\left[(\mathbf{z} - \boldsymbol{\mu})^\top\right] = 0.$$

Also, $\mathrm{E}[(\mathbf{z} - \boldsymbol{\mu})(\mathbf{z} - \boldsymbol{\mu})^\top] = \boldsymbol{\Phi}$ and $\mathrm{E}[\boldsymbol{\nu}\boldsymbol{\nu}^\top] = \boldsymbol{\Psi}$, so the covariance of $\mathbf{x}$ is $\mathbf{G}\boldsymbol{\Phi}\mathbf{G}^\top + \boldsymbol{\Psi}$.

APPENDIX B

DETAILS FOR FAST IMPLEMENTATIONS

In this appendix, we describe details needed for efficient implementations of the inference and learning algorithms for TMG. To reduce the number of variables that occur in statements about complexity, we assume that M is of order N, that is, that the size of the latent variable $\mathbf{z}$ is of the same order as the size of the observed input $\mathbf{x}$. We also assume that the number of clusters is 1, but the techniques can be applied when there is more than one cluster, since conditioning on the cluster index reduces to the same computations as when there is only one cluster.

If the transformation matrices are dense, the probabilistic inferences in Section 3 take order $|\mathcal{T}|N^3$ time. For example, computing $p(\mathbf{x}|\mathbf{T})$ for a *single* $\mathbf{T}$, requires computing the determinant of $\mathbf{T}\boldsymbol{\Phi}\mathbf{T}^\top + \boldsymbol{\Psi}$. Since this is an $N \times N$ matrix, computing its determinant takes order N^3 time.

B.1 TMG in Order $|\mathcal{T}|N$ Time Using Sparse $\mathbf{T}$

Many types of transformation (e.g., translations, rotations, shearing, and moderate scaling) can be represented using sparse transformation matrices. For example, integer-pixel translations are represented using matrices that have a single nonzero element (1) in each row. Integer-pixel translations with wrap-around are represented using permutation matrices. The inferences needed for TMG make use of computations such as multiplying transformation matrices with diagonal covariance matrices and taking inverses of such products, for each transformation. Using sparse matrix algebra, inference can be performed in order $|\mathcal{T}|N$ time.

In TMG (see Section 3), most operations are spent computing $p(\mathbf{x}|\mathbf{T}, c)$ for every $\mathbf{T}$, the posterior mean of the latent image $\mathrm{E}[\mathbf{z}|\mathbf{x}]$, the mean-squared pretransformation noise $\mathrm{E}[\mathrm{diag}((\mathbf{z} - \boldsymbol{\mu})(\mathbf{z} - \boldsymbol{\mu})^\top)|\mathbf{x}]$, and the mean-squared posttransformation noise $\mathrm{E}[\mathrm{diag}((\mathbf{x} - \mathbf{T}\mathbf{z})(\mathbf{x} - \mathbf{T}\mathbf{z})^\top)|\mathbf{x}]$.

Computing $p(\mathbf{x}|\mathbf{T})$ requires computing the determinant of $\mathbf{T}\boldsymbol{\Phi}\mathbf{T}^\top + \boldsymbol{\Psi}$ and a Mahalanobis distance of the form $(\mathbf{x} - \mathbf{T}\boldsymbol{\mu})^\top(\mathbf{T}\boldsymbol{\Phi}\mathbf{T}^\top + \boldsymbol{\Psi})^{-1}(\mathbf{x} - \mathbf{T}\boldsymbol{\mu})$. Since $\mathbf{T}$ is sparse and $\boldsymbol{\Phi}$ is diagonal, for each transformation, $\mathbf{T}\boldsymbol{\Phi}\mathbf{T}^\top + \boldsymbol{\Psi}$, its determinant, its inverse, and the above Mahalanobis distance can be computed in order N time. (In fact, if $\mathbf{T}$ has only one nonzero entry in each row such as for integer-pixel translations, $\mathbf{T}\boldsymbol{\Phi}\mathbf{T}^\top + \boldsymbol{\Psi}$ is diagonal.) So, computing $p(\mathbf{x}|\mathbf{T})$ for all $\mathbf{T}$ takes order $|\mathcal{T}|N$ time.

Computing $\mathrm{E}[\mathbf{z}|\mathbf{x}]$ requires computing the covariance of $\mathbf{z}$ under $p(\mathbf{z}|\mathbf{T}, \mathbf{x})$ for all $\mathbf{T}$. The covariance is $(\boldsymbol{\Phi}^{-1} + \mathbf{T}^\top \boldsymbol{\Psi}^{-1}\mathbf{T})^{-1}$ and since $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are diagonal and $\mathbf{T}$ is sparse, this covariance matrix can be computed in order N time. $\mathrm{E}[\mathbf{z}|\mathbf{x}]$ is given by averaging the product of this sparse

covariance matrix with $(\mathbf{\Phi}^{-1}\boldsymbol{\mu} + \mathbf{T}^\top\mathbf{\Psi}^{-1}\mathbf{x})$, over all $\mathbf{T}$. Computing $\mathbf{T}^\top\mathbf{\Psi}^{-1}\mathbf{x}$ and $\mathbf{\Phi}^{-1}\boldsymbol{\mu}$ takes order N time and multiplying by the sparse covariance matrix also takes order N time. So, computing $\mathrm{E}[\mathbf{z}|\mathbf{x}]$ takes order $|\mathcal{T}|N$ time.

As shown in Sections 3.2.3 and 3.2.5, $\mathrm{E}[\mathrm{diag}((\mathbf{z}-\boldsymbol{\mu})(\mathbf{z}-\boldsymbol{\mu})^\top)|\mathbf{x}]$ and $\mathrm{E}[\mathrm{diag}((\mathbf{x}-\mathbf{T}\mathbf{z})(\mathbf{x}-\mathbf{T}\mathbf{z})^\top)|\mathbf{x}]$ are obtained by taking element-wise sums and products of vectors (note that for vectors $\mathbf{u}$ and $\mathbf{v}$, $\mathrm{diag}(\mathbf{u}\mathbf{v}^\top)$ is just the element-wise product of $\mathbf{u}$ and $\mathbf{v}$), by computing the diagonal of the posterior covariance matrix $\mathrm{COV}[\mathbf{z}|\mathbf{T},\mathbf{x}]$, by taking products of $\mathbf{T}$ with vectors and sparse matrices, and by summing over $\mathbf{T}$. Any one term in the sum takes order N time because the matrices are sparse and the sum takes order $|\mathcal{T}|$ time. So, the mean-squared deviations can be computed in order $|\mathcal{T}|N$ time.

B.2 TMG in Order $|\mathcal{T}|\ln N$ Time Using the FFT

Often, the input $\mathbf{x}$ is measured on a discrete coordinate system where the set of transformations is all possible discrete *shifts*, with wrap-around. In this case, probabilistic inference and learning in the TMG can be significantly sped up by using the fast Fourier transform (FFT), *if we assume that the post-transformation noise is constant*, i.e., $\mathbf{\Psi} = \psi\mathbf{I}$. For example, our current 1GHz PIII MATLAB implementation of TMG can perform the E step and M step for two clusters and all possible integer-pixel x-y translations in a 320×240 image, at *15 frames per second*.

For images measured on a 2D rectangular grid, an x-y translation corresponds to a shift in the coordinate system. For images measured on a 2D radial grid, a scale and a rotation corresponds to a radial shift and an angular shift in the coordinate system [11]. To account for translations, scales, *and* rotations, a computationally efficient, but approximate, variational technique can be used to decouple inference into two sets of 2D inferences [11].

Three observations lead to $|\mathcal{T}|\ln N$ time complexity. First, when $\mathbf{\Psi} = \psi\mathbf{I}$, pre and post-transformation noise are not distinguishable, so $\mathbf{T}$ will drop from the covariance matrices in the expressions for inference. Also, the posterior covariance matrices are diagonal, since integer-pixel translations preserve the independence of pixel noise. Second, for integer-pixel shifts, the posterior probabilities of the transformations are given by convolving the input image with the mean of the latent image and the expectations of the latent image and squared noise are given by convolving the posterior probabilities with shifted versions of the input. Third, the FFT can be used to compute a convolution in order $|\mathcal{T}|\ln N$ time (a well-known fact).

We now describe techniques for computing the following inferences for TMG (see Section 3) in $|\mathcal{T}|\ln N$ time: $P(\mathbf{x}|\mathbf{T})$ for all $\mathbf{T}$, $\mathrm{E}[\mathbf{z}|\mathbf{x}]$, $\mathrm{E}[\mathrm{diag}((\mathbf{z}-\boldsymbol{\mu})(\mathbf{z}-\boldsymbol{\mu})^\top)|\mathbf{x}]$ and $\mathrm{E}[\mathrm{diag}((\mathbf{x}-\mathbf{T}\mathbf{z})(\mathbf{x}-\mathbf{T}\mathbf{z})^\top)|\mathbf{x}]$. These are the inferences that are needed to perform the M step of the EM algorithm.

First, we switch to a notation that simplifies the expression for a transformation that is a shift in the input coordinates. Let $\mathbf{i}$ be an integer vector in the discrete coordinate system on which the data is measured. For example, if $\mathbf{x}$ contains the pixel intensities in a two-dimensional $n \times n$ image ($N = n^2$), $\mathbf{i} \in \{(i_1, i_2) : i_1 = 1, \ldots, n, i_2 = 1, \ldots, n\}$. Let $x(\mathbf{i})$ be the element in the observed image corresponding to the pixel at coordinate $\mathbf{i}$ and let $z(\mathbf{i})$ and $\mu(\mathbf{i})$ be defined the same way, for the latent image and the cluster mean. Let $\phi(\mathbf{i})$ be the element on the diagonal of $\mathbf{\Phi}$ corresponding to the pixel at coordinate $\mathbf{i}$.

Using this notation, a transformation corresponding to a shift can be represented by an integer vector $\mathbf{T}$ in the same discrete coordinate system. So, the observed value $x(\mathbf{i})$, is determined by the latent value $z(\mathbf{i}+\mathbf{T})$, where "+" is taken *modulus* the boundary of the coordinate system. For an $n \times n$ image ($|\mathcal{T}| = n^2$), $\mathbf{i}+\mathbf{T} = (i_1 + T_1 \bmod n, i_2 + T_2 \bmod n)$.

The transformation likelihood from Section 3 can be written

$$p(\mathbf{x}|\mathbf{T}) = \prod_{\mathbf{i}} \frac{1}{(2\pi(\phi(\mathbf{i}+\mathbf{T})+\psi))^{1/2}} \exp\left(-\frac{(x(\mathbf{i})-\mu(\mathbf{i}+\mathbf{T}))^2}{2(\phi(\mathbf{i}+\mathbf{T})+\psi)}\right).$$

Taking the logarithm and expanding the square, we obtain

$$\begin{aligned}\ln p(\mathbf{x}|\mathbf{T}) &= -\frac{1}{2}\sum_{\mathbf{i}} \ln(2\pi(\phi(\mathbf{i}+\mathbf{T})+\psi)) \\ &- \frac{1}{2}\sum_{\mathbf{i}} x(\mathbf{i})^2\left[\frac{1}{(\phi(\mathbf{i}+\mathbf{T})+\psi)}\right] \\ &+ \sum_{\mathbf{i}} x(\mathbf{i})\left[\frac{\mu(\mathbf{i}+\mathbf{T})}{\phi(\mathbf{i}+\mathbf{T})+\psi}\right] - \frac{1}{2}\sum_{\mathbf{i}}\left[\frac{\mu(\mathbf{i}+\mathbf{T})^2}{(\phi(\mathbf{i}+\mathbf{T})+\psi)}\right].\end{aligned}$$

The first and last sums in the above expression do not depend on $\mathbf{T}$, so they can be computed in order N time. The second and third sums in the above expression have the form of a convolution

$$f(\mathbf{T}) = \sum_{\mathbf{i}} g(\mathbf{i})h(\mathbf{i}+\mathbf{T}).$$

Computing the convolution directly for all $\mathbf{T}$ takes order $|\mathcal{T}|N$ time, just like in the previous section. However, the FFT can be used to compute the convolution in order $|\mathcal{T}|\ln N$ time, as follows.

The two-dimensional FFTs $G(\boldsymbol{\omega})$ and $H(\boldsymbol{\omega})$ of g and h are computed in order $|\mathcal{T}|\ln N$ time. Then, the FFT $F(\boldsymbol{\omega})$ of f is computed in order $|\mathcal{T}|$ time by taking the element-wise product, $F(\boldsymbol{\omega}) = G(\boldsymbol{\omega})^* H(\boldsymbol{\omega})$, where "*" denotes complex conjugate. Note that the complex conjugate is needed, since $\mathbf{T}$ is *added* in the argument, not subtracted. Finally, $f(\mathbf{T})$ for all $\mathbf{T}$ is obtained by computing the inverse FFT of $F(\boldsymbol{\omega})$ in order $|\mathcal{T}|\ln N$ time.

Since $\mathbf{\Psi} = \psi\mathbf{I}$ and $\mathbf{T}^\top\mathbf{T} = \mathbf{I}$, the expected value of the latent image given $\mathbf{T}$ derived in Section simplifies to

$$\mathrm{E}[\mathbf{z}|\mathbf{T},\mathbf{x}] = \left(\mathbf{\Phi}^{-1} + \mathbf{\Psi}^{-1}\right)^{-1}\left(\mathbf{\Phi}^{-1}\boldsymbol{\mu} + \mathbf{\Psi}^{-1}\mathbf{T}^\top\mathbf{x}\right).$$

So, the expected value of the latent image simplifies to

$$\begin{aligned}\mathrm{E}[\mathbf{z}|\mathbf{x}] &= \sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}|\mathbf{x})\mathrm{E}[\mathbf{z}|\mathbf{T},\mathbf{x}] \\ &= \left(\mathbf{\Phi}^{-1} + \mathbf{\Psi}^{-1}\right)^{-1}\left(\mathbf{\Phi}^{-1}\boldsymbol{\mu} + \mathbf{\Psi}^{-1}\sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}|\mathbf{x})\mathbf{T}^\top\mathbf{x}\right).\end{aligned}$$

Using the notation described above, we have

$$\mathrm{E}[z(\mathbf{i})|\mathbf{x}] = \left(\phi(\mathbf{i})^{-1} + \psi^{-1}\right)^{-1}\left(\phi(\mathbf{i})^{-1}\mu(\mathbf{i}) + \psi^{-1}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})x(\mathbf{i}-\mathbf{T})\right).$$

The term containing the sum over $\mathbf{T}$ dominates the computation of $\mathrm{E}[z(\mathbf{i})|\mathbf{x}]$ for all $\mathbf{i}$. However, this term can

be computed for all $\mathbf{i}$ in order $|\mathcal{T}| \ln N$ time using FFTs as described above (except, the complex conjugate is not taken in this case, since $\mathbf{T}$ is subtracted from $\mathbf{i}$).

Combining expressions in Section 3.1 and using $\boldsymbol{\Psi} = \psi \mathbf{I}$ and $\mathbf{T}^\top \mathbf{T}$, the expected value of the pretransformation noise can be written

$$\begin{aligned}&\mathrm{E}\Big[\mathrm{diag}\Big((\mathbf{z}-\boldsymbol{\mu})(\mathbf{z}-\boldsymbol{\mu})^\top\Big)|\mathbf{x}\Big] \\ &\quad = \sum_{\mathbf{T}\in\mathcal{T}} P(\mathbf{T}|\mathbf{x})\mathrm{diag}((\mathrm{E}[\mathbf{z}|\mathbf{T},\mathbf{x}]-\boldsymbol{\mu}) \\ &\qquad \Big(\mathrm{E}[\mathbf{z}|\mathbf{T},\mathbf{x}]-\boldsymbol{\mu}\Big)^\top\Big) + \big(\boldsymbol{\Phi}^{-1}+\boldsymbol{\Psi}^{-1}\big)^{-1}.\end{aligned}$$

Substituting the above expression for $\mathrm{E}[\mathbf{z}|\mathbf{T},\mathbf{x}]$, simplifying, and using the new notation, we have

$$\begin{aligned}&\mathrm{E}\Big[(z(\mathbf{i})-\mu(\mathbf{i}))^2|\mathbf{x}\Big] = \Big(\phi(\mathbf{i})^{-1}+\psi^{-1}\Big)^{-1} \\ &\left(\Big(\phi(\mathbf{i})^{-1}+\psi^{-1}\Big)^{-1}\psi^{-2}\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{x})(x(\mathbf{i}-\mathbf{T})-\mu(\mathbf{i}))^2+1\right).\end{aligned}$$

The computationally intensive term is the sum over $\mathbf{T}$, which can be expanded into convolutions and computed in order $|\mathcal{T}| \ln N$ time.

By following a similar derivation as above, the computation of $\mathrm{E}[\mathrm{diag}((\mathbf{x}-\mathbf{T}\mathbf{z})(\mathbf{x}-\mathbf{T}\mathbf{z})^\top)|\mathbf{x}]$ can be performed in $|\mathcal{T}| \ln N$ time.

ACKNOWLEDGMENTS

The authors would like to thank Tom Huang for his generous support while the authors were with the Image Formation and Processing Group at the University of Illinois at Urbana-Champaign as well as Tom Minka for pointing out a useful trick for simplifying determinants of sums of products of matrices. They would also like to thank the anonymous reviewers who provided very helpful suggestions for modifying the original manuscript. Distinct portions of this research were supported separately by grants from CITO (28838), NSERC (RGPIN-217657-99), the US National Science Foundation (IRI-9634618, CDA-9624396), and a Beckman Fellow grant awarded to B.J. Frey.

REFERENCES

[1] Y. Amit and D. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Computation,* vol. 9, pp. 1545-1588, 1997.

[2] C.M. Bishop, M. Svensén, and C.K.I. Williams, "GTM: The Generative Topographic Mapping," *Neural Computation,* vol. 10, no. 1, pp. 215-235, 1998.

[3] M.J. Black, D.J. Fleet, and Y. Yacoob, Robustly Estimating Changes in Image Appearance," *Computer Vision and Image Understanding,* vol. 78, no. 1, pp. 8-31, 2000.

[4] P. Dayan and R.S. Zemel, "Competition and Multiple Cause Models," *Neural Computation,* vol. 7, pp. 565-579, 1995.

[5] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Proc. Royal Statistical Soc. B,* vol. 39, pp. 1-38, 1977.

[6] B.J. Frey, *Graphical Models for Machine Learning and Digital Communication.* Cambridge, Mass.: MIT Press, 1998.

[7] B.J. Frey and G.E. Hinton, "Variational Learning in Non-Linear Gaussian Belief Networks," *Neural Computation,* vol. 11, no. 1, pp. 193-214, 1999.

[8] B.J. Frey, G.E. Hinton, and P. Dayan, "Does the Wake-Sleep Algorithm Produce Good Density Estimators?" *Proc. Eighth Conf. Advances in Neural Information Processing Systems,* D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, eds., Dec. 1995.

[9] B.J. Frey and N. Jojic, "Estimating Mixture Models of Images and Inferring Spatial Transformations Using the EM Algorithm," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 416-422, June 1999.

[10] B.J. Frey and N. Jojic, "Transformed Component Analysis: Joint Estimation of Spatial Transformations and Image Components," *Proc. IEEE Int'l Conf. Computer Vision,* Sept. 1999.

[11] B.J. Frey and N. Jojic, "Fast, Large-Scale Transformation-Invariant Clustering," *Proc. 14th Conf. Advances in Neural Information Processing Systems,* Dec. 2001.

[12] B.J. Frey and N. Jojic, "Learning Graphical Models of Images, Videos and Their Spatial Transformations," *Proc. Uncertainty in Artificial Intelligence,* 2000.

[13] Z. Ghahramani, "Factorial Learning and the EM Algorithm," *Proc. Seventh Conf. Advances in Neural Information Processing Systems,* G. Tesauro, D. Touretzky, and T. Leen, eds., Dec. 1994.

[14] R. Golem and I. Cohen, "Scanning Electron Microscope Image Enhancement," technical report, School of Computer and Electrical Eng. Project Report, Ben-Gurion Univ., 1998.

[15] G.E. Hinton, P. Dayan, B.J. Frey, and R.M. Neal, "The Wake-Sleep Algorithm for Unsupervised Neural Networks," *Science,* vol. 268, pp. 1158-1161, 1995.

[16] G.E. Hinton, P. Dayan, and M. Revow, "Modeling the Manifolds of Images of Handwritten Digits," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, pp. 65-74, Aug. 1997.

[17] G.E. Hinton and T.J. Sejnowski, "Learning and Relearning in Boltzmann Machines," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* D.E. Rumelhart and J.L. McClelland, eds., vol. I, pp. 282-317, 1986.

[18] M. Isard and A. Blake, "Contour Tracking by Stochastic Propagation of Conditional Density," *Proc. European Conf. Computer Vision,* pp. 343-356, 1996.

[19] N. Jojic, N. Petrovic, B.J. Frey, and T.S. Huang, "Transformed Hidden Markov Models: Estimating Mixture Models of Images and Inferring Spatial Transformations in Video Sequences," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* June 2000.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE,* vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[21] D.G. Lowe, "Similarity Mmetric Learning for a Variable-Kernel Classifier," *Neural Computation,* vol. 7, no. 1, pp. 72-85, 1995.

[22] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 7, pp. 696-710, July 1997.

[23] P.Y. Simard, B. Victorri, Y. LeCun, and J. Denker, "Tangent Prop—A Formalism for Specifying Selected Invariances in an Adaptive Network," *Proc. Fourth Conf. Advances in Neural Information Processing Systems,* Dec. 1991.

[24] J.B. Tenenbaum and W.T. Freeman, "Separating Style and Content," *Proc. Ninth Conf. Advances in Neural Information Processing Systems,* M.C. Mozer, M.I. Jordan, and T. Petsche, eds., Dec. 1996.

[25] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience,* vol. 3, no. 1, 1991.

[26] N. Vasconcelos and A. Lippman, "Multiresolution Tangent Distance for Affine-Invariant Classification," *Proc. 10th Conf. Advances in Neural Information Processing Systems,* M.I. Jordan, M.I. Kearns, and S.A. Solla, eds., Dec. 1997.

[27] M. Webber, M. Welling, and P. Perona, "Towards Automatic Discovery of Object Categories," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* June 2000.

Brendan J. Frey received the doctorate degree in electrical and computer engineering from the University of Toronto in 1997, where he was an NSERC 1997 Science and Engineering Scholar and a member of Geoffrey Hinton's neural networks research group. From 1997 to 1999, he was a Beckman Fellow and an NSERC Postdoctoral Fellow at the Beckman Institute, University of Illinois at Urbana-Champaign. From 1998 to 2001, he was a faculty member in computer science at the University of Waterloo and an adjunct faculty member in electrical and computer engineering at the University of Illinois at Urbana-Champaign. Currently, he is the head of the Probabilistic and Statistical Inference Group in the Department of Electrical and Computer Engineering at the University of Toronto. He has received several awards, given more than 30 invited talks and published more than 80 papers on inference and estimation in complex probability models for machine learning, iterative decoding, computer vision, speech processing, image processing, automated diagnosis, data compression, and pattern recognition. He is an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, and is coeditor-in-chief of the February 2000 special issue of the *IEEE Transactions on Information Theory*, titled "Codes on Graphs and Iterative Algorithms." He is a member of the IEEE Computer Society.

Nebojsa Jojic received the doctoral degree in 2002 from the University of Illinois at Urbana-Champaign, where he received the Robert T. Chien Memorial Award for his research on generative models for computer vision. In addition to conducting research at the University of Illinois and Microsoft, he has consulted at the Hong Kong University of Science and Technology, and he spent a semester at the University of Illinois at Chicago. He is a researcher in the Interactive Visual Media Group at Microsoft Research. His research interests include signal processing, machine learning, computer vision, and computer graphics. He has published more than 30 papers in these areas. He is a member of the IEEE Computer Society.