Learning Appearance and Transparency Manifolds of Occluded Objects in Layers

Brendan J. Frey
University of Toronto
10 King's College Rd.
Toronto, ON, Canada

Nebojsa Jojic
Microsoft Research
One Microsoft Way
Redmond, WA, USA

Anitha Kannan
University of Toronto
10 King's College Rd.
Toronto, ON, Canada

Videos and software available at www.psi.toronto.edu/layers.html

Abstract

*By mapping a set of input images to points in a low-dimensional manifold or subspace, it is possible to efficiently account for a small number of degrees of freedom. For example, images of a person walking can be mapped to a 1-dimensional manifold that measures the phase of the person's gait. However, when the object is moving around the frame and being occluded by other objects, standard manifold modeling techniques (*e.g.*, principal components analysis, factor analysis, locally linear embedding) try to account for global motion and occlusion. We show how factor analysis can be incorporated into a generative model of layered, 2.5-dimensional vision, to jointly locate objects, resolve occlusion ambiguities, and learn models of the appearance manifolds of objects. We demonstrate the algorithm on a video consisting of four occluding objects, two of which are people who are walking, and occlude each other for most of the duration of the video. Whereas standard manifold modeling techniques fail to extract information about the gaits, the layered model successfully extracts a periodic representation of the gait of each person.*

1 Introduction

High-dimensional data such as images often lie in a much lower-dimensional manifold or subspace, that is specified by a small number of underlying degrees of freedom. In images of a person walking, the state of the walk can be represented by a single number that corresponds to the angle between the legs. In images of a face with various expressions, the state of the face can be represented by one number for each muscle in the face. Generally, the relationship between the input image and the manifold is nonlinear, but useful results have been obtained using linear mappings between the input and the manifold. A linear mapping can be found by computing the eigenvectors of the covariance matrix (a.k.a. principal components analysis) [1], or by learning a probability model called a factor analyzer [2].

Linear subspace models have been used successfully for recognizing patterns [3, 4, 5, 6, 7, 8, 9], modeling lighting [10], analyzing motion (c.f. [11] for an excellent review), tracking objects [12], jointly tracking objects and recognizing patterns [13], and analyzing human gait [14]. When the input images contain transformations such as translations, rotations, *etc.*, transformation-invariant linear subspace models can be used [15]. When the input images contain outliers, robust estimates of linear mappings can be computed [16].

In cases where linear manifold models do not suffice, they can often be modified so that they work well. If the data is locally linear, mixtures of linear models can be used [17, 18, 19], or techniques for locally linear embedding can be applied [20]. For highly nonlinear mappings, a discrete approximation can be used globally, while at the local level, the data is modeled using a linear manifold. [21] show that the subspace of gestures can be effectively discretized for the purpose of gesture recognition. [22, 23] show that the subspace of transformations (translations, rotations, *etc.*) can be efficiently discretized for the purpose of transformation-invariant clustering.

The above techniques for estimating linear subspaces are quite effective when the input images contain un-occluded examples of the objects. However, when objects occlude each other, these techniques extract subspaces that try to account for changes in appearance due to occlusion, instead of the more interesting and potentially useful subspaces that account for changes in appearance due to the underlying degrees of freedom for each object. For example, Fig. 1 shows 12 images from a 120-frame video sequence of two people walking behind a parked car and in front of a garden. In most frames of the video sequence, the man in blue-jeans is occluded (either partly or almost completely) by the man pushing the baby carriage. To properly extract the underlying degrees of freedom for each object, it is necessary to identify other, occluding objects in the scene and "subtract" them out of the image.

Although greedy techniques can be used to remove one

Figure 1: Images from a 120-frame video of 2 people walking behind a parked car and in front of a static garden.

object at a time [24], we believe that a more principled approach is to formulate a probability model of the entire scene and then perform inference in this model to jointly recover multiple objects [25]. In our probability model, the 3-dimensional scene is approximately described by a layered set of 2-dimensional appearance maps and transparency maps [26]. Previously, we showed how the mean appearance of each object in a layered model can be learned [25]. An advantage of this approach is that the scene can be "filled in" behind occlusions, using the estimated mean appearances. However, without a subspace model, the mean appearance does not account for deformations and important degrees of freedom, such as the phase of a walking person's gait, cannot be extracted.

In this paper, we describe a generative probability model and an approximate inference and learning algorithm that can *jointly* decompose an input video into a layered representation of the objects in a scene, and infer a linear subspace model of each object. Once learned, the subspace models can be used to specify the underlying degrees of freedom of each object in the scene. We compare the layered subspace model with eigen-vector based technique, on the task of extracting the phase of the gait for each of two people in the above video.

MATHEMATICAL NOTATION

If $\mathbf{x}$ and $\mathbf{y}$ are two vectors, $\mathbf{xy}$ denotes the vector given by the element-wise product of $\mathbf{x}$ and $\mathbf{y}$. If $\mathbf{x}$ is a vector, $\mathbf{x}^n$ is the vector $\mathbf{x}$ with each element raised to the power n. If $\mathbf{A}$ is a matrix and $\mathbf{x}$ is a vector, $\mathbf{A} + \mathbf{x} = \mathbf{A} + \mathrm{diag}(\mathbf{x})$, where $\mathrm{diag}(\mathbf{x})$ is the diagonal matrix with $\mathbf{x}$ on its diagonal.

2 Layered Subspace Models

Fig. 2 shows the Bayesian network for a 4-layer generative model, where the layers are arranged horizontally, and the generative steps within each layer are arranged vertically. This model was learned from the video sequence described above, using the approximate inference and learning algorithm described later in this paper. The top row of images show some model parameters, which are shared across all video frames. These include the mean appearance map and the mean transparency map for each of the 4 layers. To generate a particular video frame, a point in the subspace is randomly drawn from a Gaussian. Through a linear projection, this subspace coordinate is used to produce a deformed version of the appearance map and transparency map. In this example, the 1st and 4th layers contain static objects (the car and the background shrubbery), but the 2nd and 3rd layers contain people whose arms and legs are moving. Next, a position is drawn for each layer and the appearance and transparency maps are translated appropriately. Finally, the video frame is generated by an ordered composition of the appearance maps, using the transparency maps, where the layers are combined from right to left.

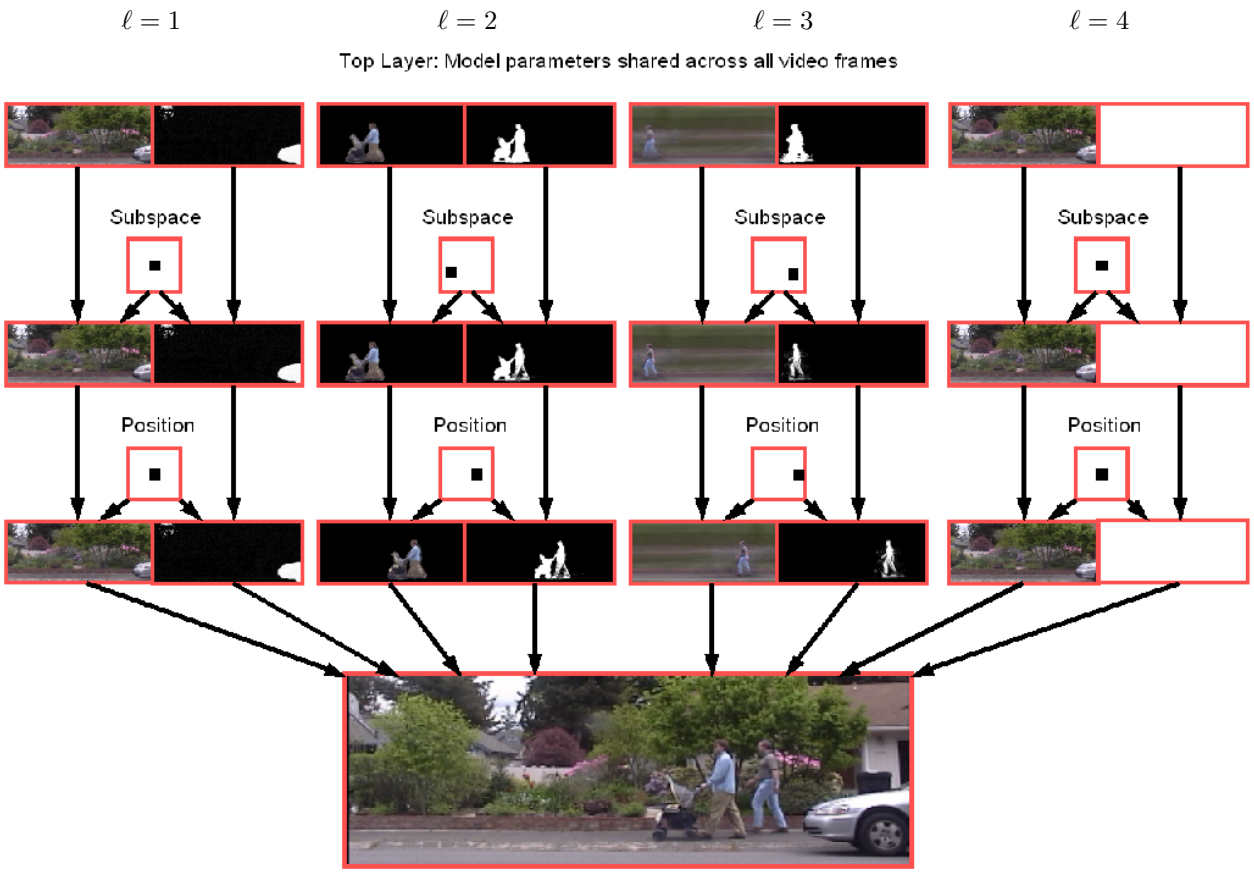

Figure 2: A Bayesian network showing how multiple subspace models (columns) combine in a layered fashion to describe a video frame. The top row of pictures shows some of the model parameters, which are the same for all video frames. The remaining pictures show the configurations of the hidden variables for the current video frame.

In a layered subspace model, the contribution from layer ℓ is specified by an appearance map vector $\mathbf{s}_\ell$ and a transparency map vector $\mathbf{m}_\ell$. Each entry in the transparency map is between 0 and 1 and specifies to what degree the corresponding pixel is transparent. When forming the input, layer ℓ contributes pixel intensities $\mathbf{m}_\ell \mathbf{s}_\ell$ (see the above section, MATHEMATICAL NOTATION). When the input is formed, some of the pixel intensities contributed by layer ℓ will be over-written by layers closer to the input. If layer i is closer to the input than layer ℓ, then the pixel intensities contributed by layer ℓ is masked by $\mathbf{1} - \mathbf{m}_i$. Taking $\ell = 1$ to be the layer that is closest to the camera and $\ell = L$ to be the layer that is farthest from the camera, the total contribution that layer ℓ makes to the input is $(\prod_{i=1}^{\ell-1}(\mathbf{1} - \mathbf{m}_i))\mathbf{m}_\ell \mathbf{s}_\ell$.

It is often convenient to allow the appearance map and transparency map to be transformed before being used to form the input For example, when combining images of objects to form an input image, it is useful to allow the image for each object to translate, rotate, *etc.* To account for transformations of the images in layer ℓ, we introduce a discrete, random transformation operator, $\mathbf{T}_\ell$ [27]. $\mathbf{T}_\ell$ can be thought of as a permutation matrix that rearranges the pixels. For example, to account for all translations in a $J \times J$ image, $\mathbf{T}_\ell$ can take on J^2 values – the permutation matrices that account for all translations. Layer ℓ contributes a transformed appearance map $\mathbf{T}_\ell \mathbf{s}_\ell$ and a transformed transparency map $\mathbf{T}_\ell \mathbf{m}_\ell$, so the total contribution that layer ℓ makes to the input is $\left(\prod_{i=1}^{\ell-1}(\mathbf{1} - \mathbf{T}_i \mathbf{m}_i)\right)\mathbf{T}_\ell \mathbf{m}_\ell \mathbf{T}_\ell \mathbf{s}_\ell$.

After the nonlinear masking operation, the contributions from all layers can be added together to approximate the input image, $\mathbf{x}$, as shown in the bottom part of Fig. 2. Assuming the approximation error is Gaussian, we have

$$p(\mathbf{x}|\{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell\}_{\ell=1}^{L}) = \mathcal{N}\left(\mathbf{x}; \sum_{\ell=1}^{L}\left(\left(\prod_{i=1}^{\ell-1}(\mathbf{1} - \mathbf{T}_i \mathbf{m}_i)\right)\mathbf{T}_\ell \mathbf{m}_\ell \mathbf{T}_\ell \mathbf{s}_\ell\right), \boldsymbol{\psi}\right), \quad (1)$$

where $\mathcal{N}(\cdot;\cdot,\cdot)$ denotes the multivariate normal density function, and $\boldsymbol{\psi}$ is a vector of pixel variances

The manifolds of appearance and transparency in each layer ℓ are modeled using a factor analyzer, whose parameters depend on the class of the object, c_ℓ. This approximates

the true manifolds using a linear subspace, plus Gaussian noise. Assuming the subspace coordinate $\mathbf{z}_\ell$ is zero-mean, unit-covariance *a priori*, the joint distribution over the appearance, transparency, transformation, class, and subspace coordinate in layer ℓ is

$$p(\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell, \mathbf{z}_\ell) = \rho_{\mathbf{T}_\ell, c_\ell} \mathcal{N}(\mathbf{s}_\ell; \boldsymbol{\mu}^s_{c_\ell} + \boldsymbol{\Lambda}^s_{c_\ell} \mathbf{z}_\ell, \boldsymbol{\phi}^s_{c_\ell}) \cdot \mathcal{N}(\mathbf{m}_\ell; \boldsymbol{\mu}^m_{c_\ell} + \boldsymbol{\Lambda}^m_{c_\ell} \mathbf{z}_\ell, \boldsymbol{\phi}^m_{c_\ell}) \mathcal{N}(\mathbf{z}_\ell; \mathbf{0}, \mathbf{I}).$$

$\rho_{\mathbf{T}_\ell, c_\ell}$ is the probability of object class c_ℓ and transformation $\mathbf{T}_\ell$. $\boldsymbol{\mu}^s_{c_\ell}$, $\boldsymbol{\Lambda}^s_{c_\ell}$ and $\boldsymbol{\phi}^s_{c_\ell}$ are the mean, factor loading matrix, and noise variances for the appearance map from class c_ℓ. $\boldsymbol{\mu}^m_{c_\ell}$, $\boldsymbol{\Lambda}^m_{c_\ell}$ and $\boldsymbol{\phi}^m_{c_\ell}$ are the same for the transparency map from class c_ℓ.

The joint distribution over all variables is given by

$$p(\mathbf{x}, \{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell, \mathbf{z}_\ell\}_{\ell=1}^L) = p(\mathbf{x} | \{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell\}_{\ell=1}^L) \prod_{\ell=1}^L p(\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell, \mathbf{z}_\ell).$$

Reasoning about the input video consists of estimating the parameters of this model, and inferring the distributions over the hidden variables. For example, if the model has been properly fit to the data, $p(\mathbf{s}_\ell | \mathbf{x})$ reveals the appearance of the object in layer ℓ, *even when the object is occluded in the input frame*. Parts of the object that are occluded will be automatically filled in using the model parameters, the inferred subspace coordinate, the inferred transformation, *etc.* As another example, $p(\mathbf{z}_\ell | \mathbf{x})$ gives the distribution over the subspace coordinate for the object in layer ℓ, and can be used, *e.g.*, to track underlying degrees of freedom, such as the gait of a walking person (see Scn. 4). In the next section, we describe an efficient inference and learning algorithm for this model.

It turns out it is sometimes useful to integrate over variables, *e.g.*, to derive inference algorithms that are more exact, as described in the next section. By integrating over the subspace coordinate, we obtain a closed-form expression for the distribution over the appearance, transparency, transformation and class for layer ℓ:

$$p(\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell) = \rho_{\mathbf{T}_\ell, c_\ell} \mathcal{N}(\mathbf{s}_\ell; \boldsymbol{\mu}^s_{c_\ell}, \boldsymbol{\Lambda}^s_{c_\ell} \boldsymbol{\Lambda}^{s\,\mathsf{T}}_{c_\ell} + \boldsymbol{\phi}^s_{c_\ell}) \cdot \mathcal{N}(\mathbf{m}_\ell; \boldsymbol{\mu}^m_{c_\ell}, \boldsymbol{\Lambda}^m_{c_\ell} \boldsymbol{\Lambda}^{m\,\mathsf{T}}_{c_\ell} + \boldsymbol{\phi}^m_{c_\ell}),$$

where "T" indicates matrix transpose. Then, the joint distribution over all variables except the subspace coordinates is given by $p(\mathbf{x}, \{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell\}_{\ell=1}^L) = p(\mathbf{x} | \{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell\}_{\ell=1}^L) \prod_{\ell=1}^L p(\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell)$.

In the above models, the variables from the different layers combine multiplicatively, and the total number of transformations is exponential in the number of layers, so exact inference in this model is intractable. Instead, we have derived several efficient, approximate inference techniques.

3 Efficient Probabilistic Reasoning, Inference and Learning

Reasoning in a hierarchical probability model consists of computing posterior distribution over hidden variables h given visible variables v, and estimating model parameters. We have explored several principled algorithms for approximate inference and learning in vision applications, including *iterative conditional modes*, *Gibbs sampling*, *variational techniques*, *structured variational techniques*, and the *sum-product algorithm* (a.k.a. *loopy belief propagation*). All of these techniques replace the intractable computation of the posterior distribution $p(h|v)$ with a search for a simplified distribution $q(h)$, that is made close to $p(h|v)$ by minimizing a "free energy":

$$\begin{aligned} F &= \int_h q(h) \log \frac{q(h)}{p(h, v)} \\ &= \int_h q(h) \log \frac{q(h)}{p(h|v)} - \log p(v) \geq -\log p(v). \end{aligned}$$

Minimizing F w.r.t. $q(h)$ minimizes the relative entropy between $q(h)$ and $p(h|v)$. Minimizing F w.r.t. $q(h)$ *and* the model parameters minimizes an upper bound on the negative log-probability of the data. See [28] for a tutorial.

The algorithms we have studied for learning manifolds of occluded objects in layers can be viewed as generalizations of our previous algorithms for transformation-invariant clustering [22, 23], transformation-invariant dimensionality reduction [29], and learning flexible sprites in video layers [25]. In the layered subspace model, given the appearance, transparency and object class in layer ℓ, the mean and variance of the subspace coordinate can be computed easily using linear algebra. So, our inference engine operates on the model described in the previous section, where the subspace coordinates are integrated out. The approximation to the posterior is

$$q(\{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell\}_{\ell=1}^L) \approx p(\{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell\}_{\ell=1}^L | \mathbf{x}),$$

where the form of q is restricted, as described below.

When selecting the form of the q-distribution for a specific problem, choices must be made about where uncertainty will be accounted for. In the layers model, a bad local optimum can occur if the inference and learning algorithm incorrectly orders the objects early on during learning and is unable to correct the mis-ordering. So, the q-distribution should account for the full joint distribution over the class labels of the objects in all layers, $C = \{c_\ell\}_{\ell=1}^L$[1]. Using point estimates for the other hidden variables, we obtain the

[1] If the number of configurations of C is too large, a factorized approximation, $q(C) = \prod_{\ell=1}^L q(c_\ell)$ can be used.

following q-distribution:

$$q(\{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell, c_\ell\}_{\ell=1}^L) =$$
$$q(C)\prod_{\ell=1}^{L}\Big(\delta(\mathbf{T}_\ell - \hat{\mathbf{T}}_{\ell,C})\delta(\mathbf{s}_\ell - \hat{\mathbf{s}}_{\ell,C})\delta(\mathbf{m}_\ell - \hat{\mathbf{m}}_{\ell,C})\Big),$$

where $\delta(\cdot)$ is the Dirac delta function. $q(C)$ is the joint distribution of the class labels of the objects in all layers. $\hat{\mathbf{T}}_{\ell,C}$, $\hat{\mathbf{s}}_{\ell,C}$ and $\hat{\mathbf{m}}_{\ell,C}$ are point estimates of the transformation, appearance and mask, given the class labels.

Given a set of model parameters, inference consists of minimizing F by setting $\mathbf{x}$ to the observed input and for each configuration of $C = \{c_\ell\}_{\ell=1}^L$, iteratively updating the estimates $\hat{\mathbf{s}}_{\ell,C}$, $\hat{\mathbf{m}}_{\ell,C}$ and $\hat{\mathbf{T}}_{\ell,C}$ until convergence. Once this is done for every configuration of C, the joint probability is used to compute $q(C)$:

$$q(C) \propto p(\mathbf{x}, \{\hat{\mathbf{s}}_{\ell,C}, \hat{\mathbf{m}}_{\ell,C}, \hat{\mathbf{T}}_{\ell,C}\}_{\ell=1}^L, C).$$

We now describe the updates for $\hat{\mathbf{s}}_{\ell,C}$, $\hat{\mathbf{m}}_{\ell,C}$ and $\hat{\mathbf{T}}_{\ell,C}$. Since these estimates are applied recursively while holding C fixed, we simplify notation by referring to them as $\mathbf{s}_\ell$, $\mathbf{m}_\ell$ and $\mathbf{T}_\ell$. The updates can be written concisely in terms of the following intermediate variables:

$$\bar{\mathbf{m}}_{1:\ell} = \prod_{i=1}^{\ell}(1 - \mathbf{T}_i\mathbf{m}_i), \quad \hat{\mathbf{x}}_{1:\ell} = \sum_{j=1}^{\ell}\bar{\mathbf{m}}_{1:j-1}\mathbf{T}_j\mathbf{m}_j\mathbf{T}_j\mathbf{s}_j,$$
$$\hat{\mathbf{x}}_{\ell:L} = \sum_{j=\ell}^{L}\frac{\bar{\mathbf{m}}_{1:j-1}}{\bar{\mathbf{m}}_{1:\ell-1}}\mathbf{T}_j\mathbf{m}_j\mathbf{T}_j\mathbf{s}_j.$$

$\bar{\mathbf{m}}_{1:\ell}$ is the composition of the inverse transparency maps from layer 1 to layer ℓ. $\hat{\mathbf{x}}_{1:\ell}$ is the estimate of the input, as given by layers 1 to ℓ. $\hat{\mathbf{x}}_{\ell:L}$ is the estimate of the input, as given by layers ℓ to L (the fraction removes the inverse transparency maps from layer 1 to layer $\ell - 1$). For any layer k, the data likelihood can be expressed as

$$p(\mathbf{x}|\{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell\}_{\ell=1}^L) =$$
$$\mathcal{N}\Big(\mathbf{x}; \hat{\mathbf{x}}_{1:k} + \bar{\mathbf{m}}_{1:k}\big(\mathbf{T}_k\mathbf{m}_k\mathbf{T}_k\mathbf{s}_k + \mathbf{T}_k(\mathbf{1} - \mathbf{m}_k)\hat{\mathbf{x}}_{k+1:L}\big), \boldsymbol{\psi}\Big),$$

where $\hat{\mathbf{x}}_{1:k}$, $\bar{\mathbf{m}}_{1:k}$ and $\hat{\mathbf{x}}_{k+1:L}$ do not depend directly on $\mathbf{s}_k$, $\mathbf{m}_k$ or $\mathbf{T}_k$. Using this form of the likelihood, it is straightforward to derive the inference updates.

The update for the transformation in the kth layer is

$$\mathbf{T}_k \leftarrow \mathrm{argmin}_{\mathbf{T}_k}\boldsymbol{\psi}^{-\mathsf{T}}\Big(\mathbf{x} - \hat{\mathbf{x}}_{1:k} -$$
$$\bar{\mathbf{m}}_{1:k}\big(\mathbf{T}_k\mathbf{m}_k\mathbf{T}_k\mathbf{s}_k + \mathbf{T}_k(\mathbf{1} - \mathbf{m}_k)\hat{\mathbf{x}}_{k+1:L}\big)\Big)^2,$$

which selects the transformation that best matches the input with the transformed appearance and transparency maps in the kth layer plus the estimate of the input from the layers above and below layer k.

The update for the appearance map in layer k is

$$\mathbf{s}_k \leftarrow \Big((\boldsymbol{\Lambda}^s_{c_k}\boldsymbol{\Lambda}^{s\,\mathsf{T}}_{c_k} + \boldsymbol{\phi}^s_{c_k})^{-1} + \mathbf{m}_k^2\mathbf{T}_k^{-1}\big(\boldsymbol{\psi}^{-1}\bar{\mathbf{m}}^2_{1:k-1}\big)\Big)^{-1}$$
$$\cdot\Big((\boldsymbol{\Lambda}^s_{c_k}\boldsymbol{\Lambda}^{s\,\mathsf{T}}_{c_k} + \boldsymbol{\phi}^s_{c_k})^{-1}\boldsymbol{\mu}^s_{c_k} + \mathbf{m}_k\mathbf{T}_k^{-1}\big(\boldsymbol{\psi}^{-1}\bar{\mathbf{m}}_{1:k-1}$$
$$\cdot(\mathbf{x} - \hat{\mathbf{x}}_{1:k-1} - \bar{\mathbf{m}}_{1:k-1}(\mathbf{1} - \mathbf{T}_k\mathbf{m}_k)\hat{\mathbf{x}}_{k+1:L})\big)\Big).$$

The appearance map is given by a weighted sum of the prior prediction $\boldsymbol{\mu}^s_{c_k}$, and the masked discrepancy between the input and the estimate of the input given by layers above and below layer k.

The update for the transparency map in layer k is

$$\mathbf{m}_k \leftarrow \Big((\boldsymbol{\Lambda}^m_{c_k}\boldsymbol{\Lambda}^{m\,\mathsf{T}}_{c_k} + \boldsymbol{\phi}^m_{c_k})^{-1} +$$
$$\mathbf{T}_k^{-1}\big(\boldsymbol{\psi}^{-1}\bar{\mathbf{m}}^2_{1:k-1}(\mathbf{T}_k\mathbf{s}_k - \hat{\mathbf{x}}_{k+1:L})^2\big)\Big)^{-1}$$
$$\cdot\Big((\boldsymbol{\Lambda}^m_{c_k}\boldsymbol{\Lambda}^{m\,\mathsf{T}}_{c_k} + \boldsymbol{\phi}^m_{c_k})^{-1}\boldsymbol{\mu}^m_{c_k} +$$
$$\mathbf{T}_k^{-1}\big(\boldsymbol{\psi}^{-1}\bar{\mathbf{m}}_{1:k-1}(\mathbf{T}_k\mathbf{s}_k - \hat{\mathbf{x}}_{k+1:L})\big)$$
$$(\mathbf{x} - \hat{\mathbf{x}}_{1:k-1} - \bar{\mathbf{m}}_{1:k-1}\hat{\mathbf{x}}_{k+1:L})\Big).$$

The transparency map is given by a weighted sum of the prior prediction $\boldsymbol{\mu}^m_{c_k}$, and the product of the data predicted in layer k with the discrepancy between the input and the estimate of the input given by layers above and below layer k.

Once probabilistic inference is complete for each video frame, the parameters are updated by minimizing F w.r.t. the model parameters. For example, $\boldsymbol{\mu}^s_i$ is set to the average value of $(\sum_C\sum_{\ell:c_\ell=i} q(C)\hat{\mathbf{s}}_{\ell,C})/(\sum_C\sum_{\ell:c_\ell=i} q(C))$, over the entire training set, where $q(C)$ weighs the filled-in appearances $\hat{\mathbf{s}}_{\ell,C}$ according to the posterior probability of the configuration of class labels, C. The updates for the factor loading matrices ($\boldsymbol{\Lambda}$'s) and object noise covariance matrices ($\boldsymbol{\phi}$'s) are similar to the standard factor analysis updates [2], except that, again, $q(C)$ is used to weigh the filled in values according to the posterior probability of the class labels.

4 Experimental Results

We trained a 4-layer model with 4 classes and a 2-dimensional manifold for each class, on a video sequence containing 120 100×250 video frames, 12 of which are shown in Fig. 1. The transformations included all horizontal and vertical translations. The parameters were initialized by setting the means to random values between 0 and 1 (the maximum pixel intensity), and setting the factor loading matrix entries to random values between -0.001 and 0.001.

Figure 3: Reconstructing the input with layer 2 deleted.

The parameters were updated 100 times, and for each parameter update, 3 iterations of the above variable updates were applied. The training time was 5 hours on a 2GHz Pentium. Fig. 2 shows the means of the appearance maps and transparency maps that were learned, along with the inferred hidden variables (appearance, transparency, subspace coordinate, position) for one video frame.

Once learned, the model can be used for analyzing the subspaces of the objects, and for visualization. For example, the transparency map for a particular layer can be set to 0, causing that layer to disappear when the input is reconstructed. Fig. 3 shows some of the input frames, along with the reconstruction with layer 3 removed. The subspace model automatically fills in the appearance of occluded parts of the image, despite deformations.

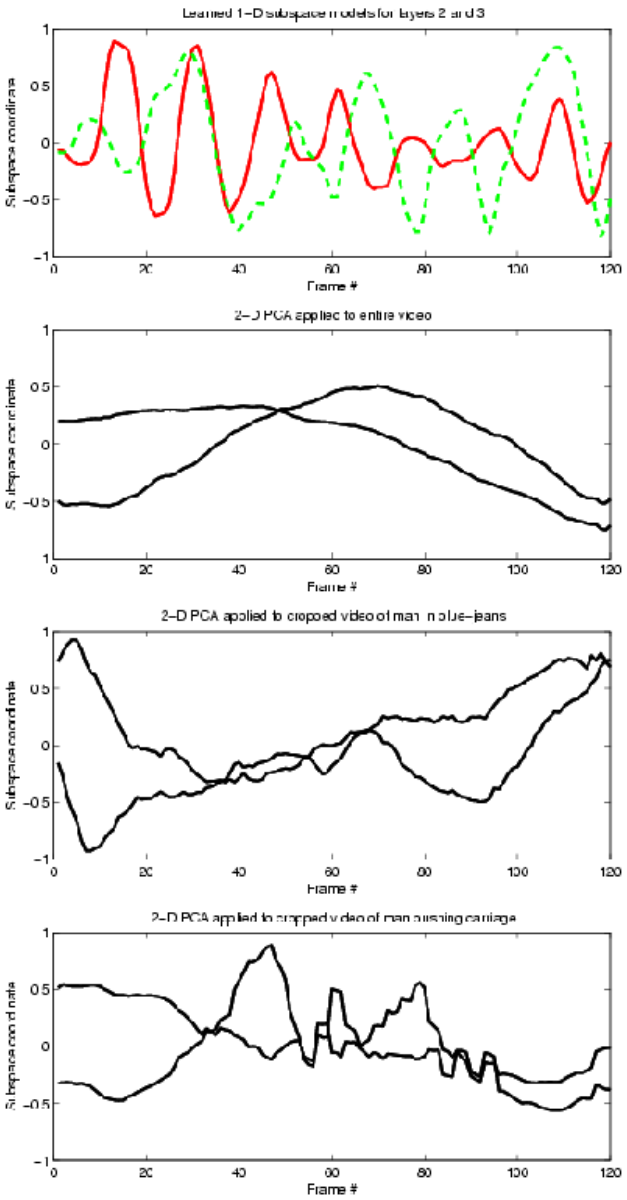

Figure 4: Top: The 1-D subspace coordinates for layer 3 (red, solid) and layer 2 (green, dashed). Second: The 2-D subspace coordinate obtained by applying PCA to the entire video sequence. Third: The 2-D subspace coordinate obtained by applying PCA to a video with the man in blue-jeans cropped by hand. Fourth: The same for the man pushing the baby carriage is tracked.

Next, we show that despite occlusions, the layered subspace model is able to recover the phase of the gait of the two people in the above video sequence. To this end, we retrained the above model using 1 dimension for each sub-

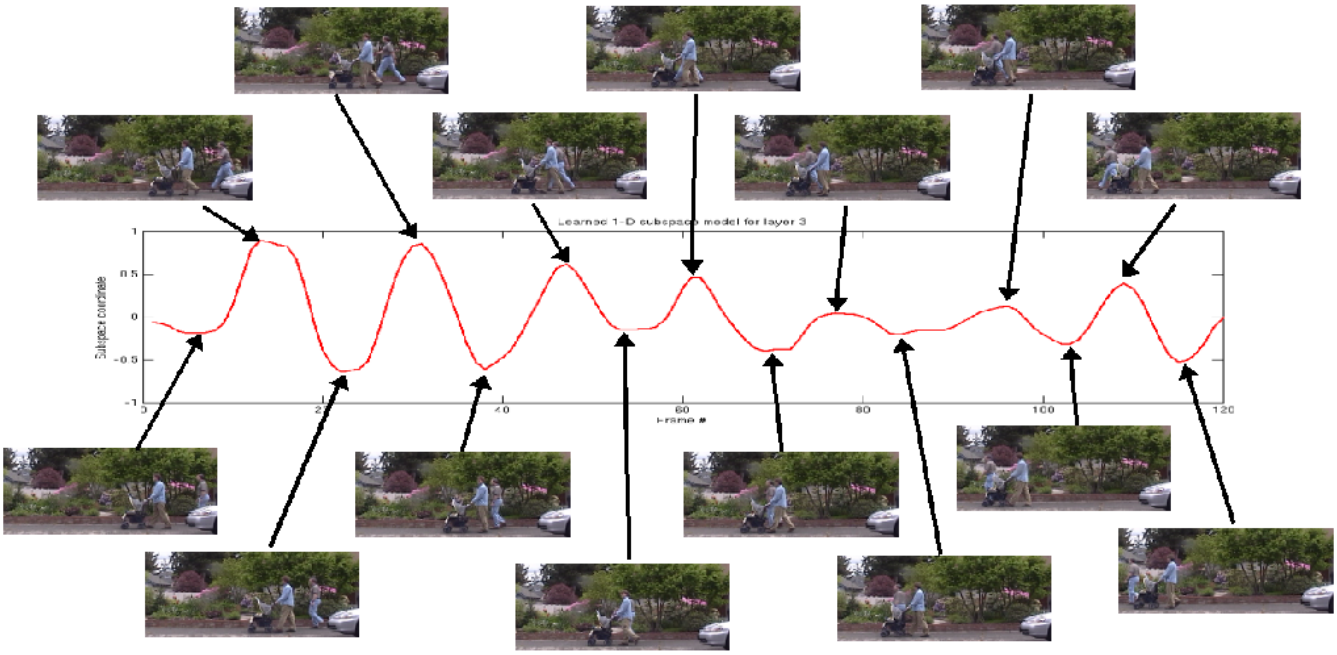

Figure 5: The input video frames corresponding to the minima and the maxima of the learned subspace coordinate for layer 3 are shown. The subspace coordinate clearly indicates the phase of the gait of the person in layer 3 (the man in blue-jeans).

space. The graph at the top of Fig. 4 shows the inferred 1-D subspace coordinate ($\mathbf{z}$) for each of the two moving layers (layers 2 and 3). The phase of the walking motion is clear for both layers, even though layer 2 occludes layer 3 for most of the duration of the video. Notice that layer 3 undergoes more cycles than layer 2, which agrees with the fact that the person in layer 3 is walking more quickly than the person in layer 2. In Fig. 4, we also compare the layered subspace model with non-layered techniques. The 2-D subspace coordinate obtained by applying PCA to the pixel-pixel covariance matrix estimated from all 120 video frames is shown in the second graph. The gait of the walkers is not evident in these signals; instead, the subspace accounts for the motion of the two walkers. We also applied standard PCA to 2 cropped videos, each of which contains a stabilized sequence of one or the other of the walkers. The bottom two graphs in Fig. 4 show the corresponding 2-D subspace coordinates. Again, the phase of the gait of each walker is not evident in these graphs.

To clearly demonstrate that the top graph in Fig. 4 has extracted the gait of the walkers, in Fig. 5 we show the video frames corresponding to the local maxima and local minima of the subspace coordinate for layer 3 (the occluded object), found by running a peak detector with width 11 video frames. It is clear that the maxima correspond to frames where the man in blue-jeans has his arm extended and his legs apart, whereas the minima correspond to frames where his arms are at his side and his legs are together.

5 Employing Deformation Fields

Instead of directly learning the linear mapping from the subspace to the image, it is often useful to use a set of basis vectors and learn the covariance of the basis vector coefficients. In [30], we describe a generative model of smoothly deformed images, using a low-frequency wavelet basis.

When the deformations are small, we can approximate the deformed image of a vector of pixel intensities $\mathbf{s}$ for an image by $\tilde{\mathbf{s}} = \mathbf{s} + (\partial\mathbf{s}/\partial x)\delta_x + (\partial\mathbf{s}/\partial y)\delta_y$, where $[\delta_x, \delta_y]$ is a deformation field (a vector field that specifies where to shift pixel intensity), and $\partial\mathbf{s}/\partial x$ and $\partial\mathbf{s}/\partial y$ are the gradient images w.r.t. horizontal and vertical shift respectively.

A smooth deformation field can be constructed from low-frequency wavelets: $\delta_x = \mathbf{R}\mathbf{z}_x$, $\delta_y = \mathbf{R}\mathbf{z}_y$, where $\mathbf{z} = [\mathbf{z}_x; \mathbf{z}_y]$ are the deformation coefficients, and columns of $\mathbf{R}$ are low-frequency wavelet basis vectors. Approximating the derivatives described above by sparse matrices $\mathbf{G}_x$ and $\mathbf{G}_y$ that operate on an input image to compute finite differences, we can express $\tilde{\mathbf{s}}$ as $\tilde{\mathbf{s}} = \mathbf{s} + (\mathbf{G}_x\mathbf{s})(\mathbf{R}\mathbf{z}_x) + (\mathbf{G}_x\mathbf{s})(\mathbf{R}\mathbf{z}_y)$, or $\tilde{\mathbf{s}} = \mathbf{s} + \Lambda(\mathbf{s})\mathbf{z}$, where $\Lambda(\mathbf{s}) = [\mathrm{diag}(\mathbf{G}_x\mathbf{s})\mathbf{R} \quad \mathrm{diag}(\mathbf{G}_y\mathbf{s})\mathbf{R}]$. This is similar to the factor analysis model used in the layered subspace model described above, where $\Lambda(\mathbf{s})$ corresponds to the factor loading matrix. Algorithms for probabilistic inference and learning in layered models of deformation fields can be derived by taking into account the fact that the factor loading matrix Λ is a function of the latent appearance map or transparency map.

6 Summary

Linear subspace models have been shown to be very effective for extracting low-dimensional manifolds of appearances of objects in images. However, when objects occlude one-another, standard subspace modeling techniques fail. We described a way to place a standard technique for linear subspace modeling, factor analysis, into a layered generative model of 2.5-dimensional vision that accounts for occlusion. Importantly, because our approach is to avoid bottom-up hacks and instead present a clearly formulated probability model, other probabilistic techniques for modeling manifolds can be incorporated into our model, *e.g.* generative topographic maps [31] and probabilistic versions of bilinear models .

Exact probabilistic inference and learning in this model is intractable, due to the nonlinear interactions between the transparency maps (masks) and appearance maps, and also due to the huge number of possible configurations of class labels and transformations associated with the objects in all layers. We presented an efficient, approximate inference and learning technique that minimizes the relative entropy (Kullback-Leibler divergence) between the true posterior distribution and an approximation to the true posterior, and maximizes a strict lower bound on the log-likelihood of the data. We showed that this algorithm can be used to extract separate manifolds of multiple, deforming objects, even when they occlude each other.

References

[1] I. T. Jolliffe, *Principal Component Analysis*, Springer-Verlag, New York NY., 1986.

[2] D. Rubin and D. Thayer, "EM algorithms for ML factor analysis," *Psychometrika*, vol. 47, no. 1, pp. 69–76, 1982.

[3] M. Turk and A Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, 1991.

[4] C. Bregler and S. M. Omohundro, "Surface learning with applications to lip-reading," in *Advances in Neural Information Processing Systems 6*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., pp. 43–50. Morgan Kaufmann, San Francisco CA., 1994.

[5] S. K. Nayar, S. Baker, and H. Murase, "Parametric feature detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[6] P. Hallinan, *A deformable model for the recognition of human faces under arbitrary illumination*, Harvard University, 1995, Doctoral dissertation.

[7] C. Nastar, B. Moghaddam, and A. Pentland, "Generalized image matching: Statistical learning of physically-based deformations," in *Proceedings of the European Conference on Computer Vision*, 1996.

[8] D. beymer, "Feature correspondence by interleaving shape and texture computations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[9] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696–710, July 1997.

[10] G. Hager and P. Belhumeur, "Real-time tracking of image regions with changes in geometry and illumination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[11] D. J. Fleet, M. J. Black, and Y. Yacoob, "Design and use of linear models for image motion analysis," *International Journal of Computer Vision*, vol. 36, no. 3, pp. 171–193, 2000.

[12] A. Blake and M. Isard, *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*, Springer-Verlag, New York NY., 1998.

[13] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[14] C. BenAbdelkader, R. Cutler, and L. Davis, "Motion-based recognition of people in eigen-gait space," in *Proceedings of the Fifth International Conference on Automatic Face and Gesture Recognition*, 2002.

[15] A. Kannan, N. Jojic, and B. J. Frey, "Fast transformation-invariant factor analysis," in *Advances in Neural Information Processing Systems 2002, Volume 15*. MIT Press, Cambridge MA., 2003.

[16] F. De la Torre and M. J. Black, "A framework for robust subspace learning," *International Journal on Computer Vision*, 2002.

[17] G. E. Hinton, M. Revow, and P. Dayan, "Recognizing handwritten digits using mixtures of linear models," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds. 1995, pp. 1015–1022, MIT Press.

[18] Z. Ghahramani and G. E. Hinton, "The EM algorithm for mixtures of factor analyzers," University of Toronto Technical Report CRG-TR-96-1, 1997.

[19] B. J. Frey, A. Colmenarez, and T. S. Huang, "Mixtures of local linear subspaces for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1998, pp. 32–37.

[20] S. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2001.

[21] A. Wilson and A. Bobick, "Parametric hidden Markov models for gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, 1999.

[22] B. J. Frey and N. Jojic, "Estimating mixture models of images and inferring spatial transformations using the EM algorithm," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1999, pp. 416–422.

[23] B. J. Frey and N. Jojic, "Transformation-invariant clustering using the EM algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, 2003.

[24] C. W. Williams and M. K. Titsias, "Learning about multiple objects in images: Factorial learning without factorial search," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, Cambridge MA., 2003.

[25] N. Jojic and B. J. Frey, "Learning flexible sprites in video layers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[26] E. H. Adelson and P. Anandan, "Ordinal characteristics of transparency," in *Proceedings of AAAI Workshop on Qualitative Vision*, 1990.

[27] B. J. Frey and N. Jojic, "Fast, large-scale transformation-invariant clustering," in *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge MA., 2001.

[28] B. J. Frey and N. Jojic, "Advances in algorithms for inference and learning in complex probability models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003, to appear as an invited tutorial paper.

[29] B. J. Frey and N. Jojic, "Transformed component analysis: Joint estimation of spatial transformations and image components," in *Proceedings of the IEEE International Conference on Computer Vision*, September 1999.

[30] N. Jojic, P. Simard, B. J. Frey, and D. Heckerman, "Separating appearance from deformation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2001.

[31] 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.

[32] J. B. Tenenbaum and W. T. Freeman, "Separating style and content," in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. MIT Press, Cambridge MA., 1997.

[33] B. J. Frey, A. Kannan, and N. Jojic, "Product analysis: Learning to model observations as products of hidden variables," in *Advances in Neural Information Processing Systems 14*, S. Becker T. G. Dietterich and Z. Ghahraman, Eds. MIT Press, Cambridge MA., 2002.