LETTER Communicated by Bruno Olshausen

Learning the Lie Groups of Visual Invariance

Xu Miao
xm@cs.washington.edu
Rajesh P. N. Rao
rao@cs.washington.edu
Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195, U.S.A.

A fundamental problem in biological and machine vision is visual invariance: How are objects perceived to be the same despite transformations such as translations, rotations, and scaling? In this letter, we describe a new, unsupervised approach to learning invariances based on Lie group theory. Unlike traditional approaches that sacrifice information about transformations to achieve invariance, the Lie group approach explicitly models the effects of transformations in images. As a result, estimates of transformations are available for other purposes, such as pose estimation and visuomotor control. Previous approaches based on first-order Taylor series expansions of images can be regarded as special cases of the Lie group approach, which utilizes a matrix-exponential-based generative model of images and can handle arbitrarily large transformations. We present an unsupervised expectation-maximization algorithm for learning Lie transformation operators directly from image data containing examples of transformations. Our experimental results show that the Lie operators learned by the algorithm from an artificial data set containing six types of affine transformations closely match the analytically predicted affine operators. We then demonstrate that the algorithm can also recover novel transformation operators from natural image sequences. We conclude by showing that the learned operators can be used to both generate and estimate transformations in images, thereby providing a basis for achieving visual invariance.

1 Introduction

How does the visual system recognize objects despite transformations such as translations, rotations, and scaling? J. J. Gibson, an early pioneer in vision research, hypothesized that "constant perception depends on the ability of the individual to detect the invariants" (Gibson, 1966). One of the first computational approaches to perceptual invariance was proposed by Pitts and McCulloch in their article, "How We Know Universals" (Pitts & McCulloch, 1947). Pitts and McCulloch suggested that invariance could be achieved by

representing an image using characteristic values, such as averages over all transformations of functionals applied to transformed images. The method, however, suffered from the drawback that it required very large numbers of neurons for computing transformed images and their functional values. A number of other approaches have since been explored (Fukushima, 1980; Hinton, 1987; LeCun et al., 1989; Olshausen, Anderson, & Essen, 1995; Frey & Jojic, 1999; Tenenbaum & Freeman, 2000; Vasilescu & Terzopoulos, 2002; Grimes & Rao, 2005), some relying on pooling of activities in a feature-detector hierarchy (e.g., Fukushima, 1980), others relying on temporal sequences of input patterns undergoing transformations (e.g., Földiák, 1991; Wiskott & Sejnowski, 2002), and yet others utilizing modifications to the distance metric for comparing input images to stored templates (e.g., Simard, LeCun, & Denker, 1993; Vasconcelos & Lippman, 2005). In a majority of these approaches, information regarding the underlying transformations is lost in the process of achieving invariance.

In this letter, we investigate a new approach to invariance based on explicitly modeling transformations in images. Transformations are modeled using operators that are learned directly from input images. Once learned, the operators can be used to achieve visual invariance by factoring out any transformation-induced changes in the image, thereby preserving the original image. Information regarding transformations is retained and can be used for tasks such as pose estimation, visuomotor planning, and control.

Our approach is based on the notion of continuous transformations and Lie group theory. It generalizes previous approaches based on first-order Taylor series expansions of images (Black & Jepson, 1996; Rao & Ballard, 1998), which can account for only small transformations due to their assumption of a linear generative model for the transformed images. The Lie approach, on the other hand, utilizes a matrix-exponential-based generative model that can handle arbitrarily large transformations once the correct transformation operators have been learned. Although Lie groups have previously been used in visual perception (Dodwell, 1983), computer vision (Van Gool, Moons, Pauwels, & Oosterlinck, 1995), and image processing (Nordberg, 1994), the question of whether it is possible to learn these groups directly from input data has remained open.

The ability to learn transformation operators from data is important because it opens the door to learning new operators that cannot be easily characterized analytically (e.g., nonrigid transformations such as those induced by changes in facial expression). The problem of learning transformations is made difficult by the fact that not only are the transformations operators unknown, but so is the amount of transformation between any given pair of images.

We present an expectation-maximization-based (EM) (Dempster, Laird, & Rubin, 1977) unsupervised learning algorithm wherein the transformation values are treated as hidden (latent) variables and the transformation operators are treated as parameters to be estimated in the M step. This

method is a natural extension of a previous gradient-descent-based algorithm proposed in Rao and Ruderman (1999). We investigate the performance of the proposed learning method on the problem of learning Lie operators for affine image transformations from an artificially constructed data set consisting of image pairs containing up to six types of transformations (translation, rotation, scaling, and two types of shear). Our experimental results demonstrate that the EM-based method has better convergence properties than the gradient-descent-based method and can accurately learn the Lie group operators for all six types of affine transformations in a completely unsupervised fashion.

A major contribution of this letter is the demonstration that for cases where the transformation operators are known, the proposed EM-based algorithm does indeed produce exactly these operators, thereby opening the door to using the algorithm in other domains where the operators are unknown. As an example, we show that the algorithm can learn a novel Lie transformation operator from a natural image sequence. We conclude by demonstrating that the learned operators can be used to estimate multiple simultaneous transformations in images for invariant recognition.

2 Continuous Transformations and Lie Groups

Suppose we have a point (in general, a vector) I_0, which is an element in a space F. Let TI_0 denote a transformation of the point I_0 to another point, say I_1. The transformation operator T is completely specified by its actions on all points in the space F. Suppose T belongs to a family of operators $\mathcal{T}$. Consider the case where $\mathcal{T}$ is a group, that is, there exists a mapping $f : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$ from pairs of transformations to another transformation such that (1) f is associative, (2) there exists a unique identity transformation, and (3) for every $T \in \mathcal{T}$, there exists a unique inverse transformation of T. We are interested in transformation groups because most common types of image transformation obey properties 1 to 3. For example, it is easy to see that translation is associative $(T_1 \cdot (T_2 \cdot T_3)I_0 = (T_1 \cdot T_2) \cdot T_3 I_0)$, with a unique identity (zero translation) and a unique inverse (the inverse of T is $-T$).

Continuous transformations are those that can be made infinitesimally small. Due to their favorable properties as described below, we will be especially concerned with continuous transformation groups or Lie groups. Continuity is associated with both the transformation operators T and the group $\mathcal{T}$. Each $T \in \mathcal{T}$ is assumed to implement a continuous mapping from $F \rightarrow F$. We focus on the case where T is parameterized by a single real number z (multiple transformations in an image can be handled by combining several single-parameter transformations as discussed below). Then the group $\mathcal{T}$ is continuous if the function $T(z) : \Re \rightarrow \mathcal{T}$ is continuous; that is, any $T \in \mathcal{T}$ is the image of some $z \in \Re$, and any continuous variation of z results in a continuous variation of T. Let $T(0)$ be equivalent to the

identity transformation. Then as $z \to 0$, the transformation $T(z)$ gets arbitrarily close to identity. Its effect on I_0 can be written as (to first order in z) $T(z)I_0 \approx (1 + zG)I_0$ for some matrix G, which is known as the generator (or operator) for the transformation group. A macroscopic transformation $I_1 = I(z) = T(z)I_0$ can be produced by chaining together a number of these infinitesimal transformations. For example, by dividing the parameter z into N equal parts and performing each transformation in turn, we obtain

$$I(z) = (1 + (z/N)G)^N I_0. \tag{2.1}$$

In the limit $N \to \infty$, this expression reduces to the matrix exponential equation,

$$I(z) = e^{zG} I_0, \tag{2.2}$$

where I_0 is the initial or reference input. Thus, each of the elements of our one-parameter Lie group can be written as $T(z) = e^{zG}$. The generator G of the Lie group is related to the derivative of $T(z)$ with respect to z: $\frac{d}{dz}T = GT$.[1] This leads to an alternate way of deriving equation 2.2. Consider the Taylor series expansion of a transformed input $I(z)$ in terms of a previous input $I(0)$:

$$I(z) = I(0) + \frac{dI(0)}{dz}z + \frac{d^2I(0)}{dz^2}\frac{z^2}{2} + \ldots, \tag{2.3}$$

where z denotes the relative transformation between $I(z)$ and $I(0)$. Defining $\frac{d}{dz}I = GI$ for some operator matrix G, we can rewrite equation 2.3 as $I(z) = e^{zG}I_0$, which is the same as equation 2.2 with $I_0 = I(0)$. Thus, some previous approaches based on first-order Taylor series expansions (Shi & Tomasi, 1994; Black & Jepson, 1996; Rao & Ballard, 1998) can be viewed as special cases of the Lie group–based generative model.

3 Learning Lie Transformation Groups

In this section, we address the following problem: Can one learn the generators (or operators) G of particular Lie transformation groups directly from input data containing examples of transformations? Note that learning the operator for a transformation effectively allows us to remain invariant to that transformation because the effects of transformations can be modeled independent of the content of the image (see below for details). We assume

[1] The generator G of a Lie group defines a field of tangent vectors and is in fact an element of the Lie algebra associated with the Lie group. We refer interested readers to Helgason (2001) for more details on the relationship between Lie algebras and Lie groups.

that during natural temporal sequences of images containing transformations, there are small image changes corresponding to deterministic sets of pixel changes that are independent of what the actual pixel values are. The rearrangements themselves are universal as in, for example, image translations. The question we address is: Can we learn the Lie group operator G given simply a series of before and after images?

Let the $n \times 1$ vector $\mathbf{I}(0)$ be the original reference image, and let $\mathbf{I}(\mathbf{z})$ be the transformed image. Each element z_i of the $T \times 1$ transformation vector $\mathbf{z}$ represents the amount of transformation of type i present in the new image. Although the methods we describe are not limited to a particular type of transformation, for concreteness, this letter focuses on the well-known group GA(2) of general affine transformations in 2D obtained under the condition of weak perspective viewing. Our focus on this group is motivated by the fact that it includes the most common types of image transformations: translations in X ($\frac{\partial}{\partial x}$) and Y ($\frac{\partial}{\partial y}$), rotations ($-y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}$), scaling ($x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y}$), and two types of hyperbolic deformations (parallel hyperbolic deformation along X/Y $x\frac{\partial}{\partial x} - y\frac{\partial}{\partial y}$ and hyperbolic deformation along the diagonals $y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}$) that can model image distortion under weak perspective viewing.

Based on the derivation in the previous section, we propose the following statistical generative model for images:

$$\mathbf{I}(\mathbf{z}) = e^{\sum_i z_i G_i}\mathbf{I}(0) + \mathbf{n}, \tag{3.1}$$

where G_i is the Lie operator matrix for transformation type i and $\mathbf{n}$ is a zero-mean gaussian white noise process with variance σ^2. Note that this generalizes equation 2.2 to the case of multiple transformations. The original image $\mathbf{I}(0)$ can itself be modeled using, for example, the following commonly used linear generative model:

$$\mathbf{I}(0) = U\mathbf{r} + \mathbf{n}_0, \tag{3.2}$$

where U is a matrix of learned object features (or basis vectors) and $\mathbf{r}$ is the feature vector (basis coefficients). Traditional approaches to appearance-based vision (such as the eigenfaces method) have focused on the generative model in equation 3.2. Equation 3.2 also forms the basis for image coding techniques such as sparse coding (Olshausen & Field, 1996) and independent component analysis (ICA; Bell & Sejnowski, 1995). None of these techniques addresses the problem of image transformations. The matrix-exponential generative model in equation 3.1 extends these previous techniques by explicitly including transformations in the generative model, independent of the actual basis matrix U used to model $\mathbf{I}(0)$. For the rest of the letter, we assume an arbitrary model for $\mathbf{I}(0)$ and focus on the problem of learning and estimating transformations on $\mathbf{I}(0)$ based on equation 3.1.

In the case where the transformations are infinitesimal, the higher-order terms become negligible, and we can rewrite equation 3.1 as

$$\Delta \mathbf{I} = \sum_i z_i G_i \mathbf{I}(0) + \mathbf{n}, \tag{3.3}$$

where $\Delta \mathbf{I} = \mathbf{I}(\mathbf{z}) - \mathbf{I}(0)$ is the difference image. Note that although this model is linear, the operators G_i learned using infinitesimal transformations are the same as those used in the exponential model. Thus, once learned, these matrices can be used to handle larger transformations as well, as explored in section 4.

Given the generative model in equation 3.3, our goal is to learn various Lie operators G_i directly from image pairs containing examples of transformations. The problem is complicated by the fact that the transformation values z_i are also unknown for each pair of images.

We first briefly review a previous gradient-descent-based approach (Rao & Ruderman, 1999) for tackling this problem and then present our expectation-maximization-based approach in section 3.2.

3.1 Approximate Method based on Gradient Descent. Suppose we are given M image pairs as data. We wish to find the $n \times n$ matrices G_i and the transformation vectors $\mathbf{z}$ that generated the data set. To do so, we take a Bayesian maximum a posteriori approach using gaussian priors on $\mathbf{z}$ and G_i. Based on equation 3.3, the negative log posterior probability of a single image pair $\{\mathbf{I}(\mathbf{z}), \mathbf{I}(0)\}$ can be written as

$$\begin{aligned} E(G_i, \mathbf{z}) &= -\log p[G_i, \mathbf{z}|\mathbf{I}(\mathbf{z}), \mathbf{I}(0)] \\ &= \frac{1}{2\sigma^2} \left\| \Delta \mathbf{I} - \sum_i z_i G_i \mathbf{I}(0) \right\|^2 \\ &\quad + \frac{1}{2\sigma_z^2} \sum_i z_i^2 + \frac{1}{2\sigma_g^2} \sum_i \mathbf{g_i}^T \mathbf{g_i}, \end{aligned} \tag{3.4}$$

where $||\cdot||$ denotes Euclidean norm, σ_z^2 is the variance of the zero-mean gaussian priors associated with the components z_i of the vector $\mathbf{z}$, $\mathbf{g_i}$ is the $n^2 \times 1$ vector form of G_i, and σ_g^2 is the variance associated with the gaussian prior on G_i. Minimizing E is equivalent to maximizing the posterior probability $P[G_i, \mathbf{z}|\mathbf{I}(\mathbf{z}), \mathbf{I}(0)]$. The posterior probability for the entire data set can be expressed as a sum of E over all M image pairs.

If the transformation values z_i are known for each image pair, the $n \times n$ operator matrices G_i can be estimated by performing gradient descent on E with respect to each G_i:

$$\dot{G}_i = -\alpha \frac{\partial E}{\partial G_i} = \frac{\alpha}{\sigma^2} \left(\Delta \mathbf{I} - \sum_i z_i G_i \mathbf{I}(0) \right) (z_i \mathbf{I}(0))^T - \frac{\alpha}{\sigma_g^2} G_i, \tag{3.5}$$

where α is a positive constant (the learning rate).

If the z_i are not known but estimates of G_i are available, one can estimate z_i by performing gradient descent on E with respect to each z_i:

$$\dot{z}_i = -\beta \frac{\partial E}{\partial z_i} = \frac{\beta}{\sigma^2} (G_i \mathbf{I}(0))^T \left(\Delta \mathbf{I} - \sum_i z_i G_i \mathbf{I}(0) \right) - \frac{\beta}{\sigma_z^2} z_i. \tag{3.6}$$

Note that if the prior distributions on z_i are uniform rather than gaussian, one can estimate the transformation vector $\mathbf{z}$ directly using the following matrix pseudoinverse method:

$$\widehat{\mathbf{z}} = [A^T A]^{-1} A^T \Delta \mathbf{I}, \tag{3.7}$$

where

$$A = \begin{bmatrix} G_1 \mathbf{I}(0) & G_2 \mathbf{I}(0) & \cdots & G_T \mathbf{I}(0) \end{bmatrix}.$$

This estimate for $\mathbf{z}$ minimizes the squared error term (first term) in equation 3.4 for fixed G_i, as can be verified by setting the partial derivative of E with respect to $\mathbf{z}$ equal to zero and solving for $\mathbf{z}$.

In the completely unsupervised case where both z_i and G_i are unknown, one can repeat the following two steps: (1) estimate z_i for all image pairs using the values for G_i from the previous iteration, and (2) adapt G_i using these estimated values for z_i. Although there is no a priori reason for such a strategy to converge (because parameters are being optimized separately), such an approach has been successfully used in other applications such as sparse coding of natural images (Olshausen & Field, 1996) and predictive coding (Rao & Ballard, 1999). We explore the convergence properties of this approach for unsupervised transformation learning in section 4.1.1.

The estimation rules for z_i given above are based on a first-order model (see equation 3.3) and are therefore useful only for estimating small (infinitesimal) transformations. A more general rule for estimating larger transformations is obtaining by performing gradient descent on the

optimization function given by the matrix-exponential generative model in equation 3.1:

$$\dot{z}_i = \gamma \left(e^{\sum_i z_i G_i} G_i \mathbf{I}(0) \right)^T \left(\mathbf{I}(\mathbf{z}) - e^{\sum_i z_i G_i} \mathbf{I}(0) \right) - \frac{\gamma}{\sigma_z^2} z_i. \tag{3.8}$$

Note that since the optimization function for the above equation is no longer quadratic in z_i, there may exist many local minima, and the gradient descent procedure is not guaranteed to converge to a global optimum. We alleviate this problem by performing random-restart gradient descent, starting from several different initial values for z_i and picking the estimated z_i that produces the best fit to equation 3.1.

Note that each of the equations for estimating z_i minimizes an optimization function that factors a new image $\mathbf{I}(\mathbf{z})$ into a reference image $\mathbf{I}(\mathbf{0})$ and a transformation component given by $\mathbf{z}$ and G_i. When $\mathbf{z}$ and G_i are correctly estimated, any new transformations in the reference image $\mathbf{I}(\mathbf{0})$ are absorbed by the transformation component, keeping the original image stable. Thus, an object recognition system trained on reference images $\mathbf{I}(\mathbf{0})$ (e.g., eigen-based systems; see equation 3.2) will continue to recognize training objects despite the presence of transformations, thereby achieving perceptual invariance (see Rao & Ballard, 1998).

3.2 EM Algorithm for Learning Lie Operators. In the completely unsupervised case where both the transformations z_i and the operators G_i are unknown, an approach based on alternating between gradient descent over z_i and G_i was suggested above. However, such an approach is not guaranteed to converge. A more rigorous approach that is guaranteed to converge (albeit to a local minimum) is to use the expectation maximization (EM) algorithm (Dempster et al., 1977). Rather than attempting to estimate a single best value for the transformations z_i, the EM algorithm treats the z_i as hidden variables and marginalizes over these variables in the optimization function, allowing G_i to be estimated in an unsupervised manner.

3.2.1 Optimization Function. The EM algorithm maximizes the likelihood of input data $\mathbf{X}$ given a set of model parameters Θ by maximizing the following optimization function,

$$Q(\Theta, \widehat{\Theta}) = \int_Y \log p(X, Y|\Theta) p(Y|X, \widehat{\Theta}) dY,$$

where $\widehat{\Theta}$ is an estimate of the parameters and Y denotes the hidden variables. Recall the generative model used in equation 3.3 (to simplify notation,

we use $\mathbf{I}$ for $\mathbf{I}(\mathbf{z})$ and $\mathbf{I}_0$ for $\mathbf{I}(0)$):

$$\mathbf{I} = \mathbf{I}_0 + \sum_i z_i G_i \mathbf{I}_0 + \mathbf{n}, \tag{3.9}$$

where we assume $\mathbf{n}$ is a zero-mean gaussian noise process with a covariance matrix Σ. For this generative model, we obtain:

$$X = \left\{ I, I_0 | I, I_0 \in \Re^{N \times M} \right\}$$
$$Y = Z \in \Re^{T \times M}$$
$$\Theta = \{ \mathbf{G}, \Sigma | \mathbf{G} \in \Re^{N \times N \times T}, \Sigma \in \Re^{N \times N} \},$$

where I and I_0 are matrices whose columns contain training images $\mathbf{I}$ and $\mathbf{I}_0$, respectively; $N = n \times n$ is the size of the 2D image (vectorized in raster format as $\mathbf{I}$ or $\mathbf{I}_0$); M is the number of training image pairs; Z is the matrix whose columns store $\mathbf{z}$ for each training image pair; T is the number of types of transformations in the training set (size of $\mathbf{z}$); and $\mathbf{G}$ is a 3D matrix containing Lie operator matrices $G_i, i = 1, \ldots, T$.

For a given pair of training images $\{\mathbf{I}_i, \mathbf{I}_{0_i}\}$, we obtain from equation 3.9,

$$p(\mathbf{I}_i | \mathbf{I}_{0_i}, \mathbf{z}_i, \mathbf{G}, \Sigma) = \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{I}_i - \mathbf{I}_{0_i} - \Sigma_j z_{ji} G_j \mathbf{I}_{0_i})^T \Sigma^{-1} (\mathbf{I}_i - \mathbf{I}_{0_i} - \Sigma_k z_{ki} G_k \mathbf{I}_{0_i})}. \tag{3.10}$$

Using Bayes' rule and assuming that each training pair is drawn independently, we can compute the conditional distribution of the hidden variable Z:

$$p(Z | I, I_0, \widehat{\mathbf{G}}, \widehat{\Sigma}) = \prod_{i=1}^{M} \alpha_i \, p(\mathbf{I}_i | \mathbf{I}_{0_i}, \mathbf{z}_i, \widehat{\mathbf{G}}, \widehat{\Sigma}) p(\mathbf{z}_i),$$

where α_i is a normalization factor for the ith image pair, and $\widehat{\mathbf{G}}$ and $\widehat{\Sigma}$ are estimated values of $\mathbf{G}$ and Σ, respectively. The optimization function thus becomes

$$Q(\Theta, \widehat{\Theta}) = \langle \log(p(I | I_0, Z, \mathbf{G}, \Sigma) p(I_0) p(Z)) \rangle_{Z | I, I_0, \widehat{\mathbf{G}}, \widehat{\Sigma}}, \tag{3.11}$$

where $\langle \cdot \rangle_{(\cdot)}$ denotes averaging with respect to the distribution in the subscript. To simplify notation, we define

$$\langle \cdot \rangle \equiv \langle \cdot \rangle_{Z | I, I_0, \widehat{\mathbf{G}}, \widehat{\Sigma}}.$$

The optimization function then becomes:

$$\begin{aligned}
Q(\Theta, \widehat{\Theta}) = \Bigg\langle & \sum_{i=1}^{M} \left(-\frac{1}{2} \log |\Sigma| \right) \\
& -\frac{1}{2} \sum_{i=1}^{M} (\mathbf{I}_i - \mathbf{I}_{0_i} - \Sigma_j z_{ji} G_j \mathbf{I}_{0_i})^T \Sigma^{-1} (\mathbf{I}_i - \mathbf{I}_{0_i} - \Sigma_k z_{ki} G_k \mathbf{I}_{0_i}) \\
& + \sum_{i=1}^{M} \log p(\mathbf{z}_i) + const. \Bigg\rangle \\
= & -\frac{1}{2} M \log |\Sigma| - \frac{1}{2} \sum_i ((\mathbf{I}_i - \mathbf{I}_{0_i})^T \Sigma^{-1} (\mathbf{I}_i - \mathbf{I}_{0_i})) \\
& -\frac{1}{2} \sum_i \sum_j \sum_k \left(\mathbf{I}_{0_i}^T G_j^T \Sigma^{-1} G_k \mathbf{I}_{0_i} \langle z_{ji} z_{ki} \rangle \right) \\
& + \sum_i \sum_j \left((\mathbf{I}_i - \mathbf{I}_{0_i})^T \Sigma^{-1} G_j \mathbf{I}_{0_i} \langle z_{ji} \rangle \right) + \sum_i \langle \log p(\mathbf{z}_i) \rangle + const.
\end{aligned}$$

3.2.2 The Expectation Step. In the expectation step or E-step, we need to calculate $\langle \mathbf{z}_i \rangle$ and $\langle \mathbf{z}_i \mathbf{z}_i^T \rangle$. We show in the appendix that

$$\langle \mathbf{z}_i \rangle = \mu_i, \tag{3.12}$$

$$\left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle = \Sigma_{zi} + \mu_i \mu_i^T, \tag{3.13}$$

where

$$\mu_i = \Sigma_{zi}^{-1} A_i^T \Sigma^{-1} (\mathbf{I}_i - \mathbf{I}_{0_i}), \tag{3.14}$$

$$\Sigma_{zi}^{-1} = A_i^T \Sigma^{-1} A_i, \tag{3.15}$$

$$A_i = \begin{bmatrix} G_1 \mathbf{I}_{0_i} & G_2 \mathbf{I}_{0_i} & \cdots & G_T \mathbf{I}_{0_i} \end{bmatrix}.$$

3.2.3 Maximization Step. The expectations calculated in the E-step allow us to maximize Q with respect to $\mathbf{G}$ and Σ. Setting $\frac{\partial Q}{\partial G_j}$ and $\frac{\partial Q}{\partial \Sigma^{-1}}$ equal to zero, we obtain (see the appendix):

$$\begin{bmatrix} G_1 & G_2 & \cdots \end{bmatrix} = \sum_i kron(\langle \mathbf{z}_i^T \rangle, C_i) \left(\sum_i kron\left(\left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle, D_i \right) \right)^{-1}, \tag{3.16}$$

$$\begin{aligned}
\Sigma = \frac{1}{M} \Bigg(& \sum_i (\mathbf{I}_i - \mathbf{I}_{0_i})(\mathbf{I}_i - \mathbf{I}_{0_i})^T + \sum_{i,j,k} G_j \mathbf{I}_{0_i} \mathbf{I}_{0_i}^T G_k^T \langle z_{ji} z_{ki} \rangle \\
& -2 \sum_{i,j} (\mathbf{I}_i - \mathbf{I}_{0_i}) \mathbf{I}_{0_i}^T G_j^T \langle z_{ji} \rangle \Bigg),
\end{aligned} \tag{3.17}$$

where *kron* denotes the Kronecker product (or tensor product) (Eves, 1980) (see the appendix for definition), $C_i = (\mathbf{I}_i - \mathbf{I}_{0_i})\mathbf{I}_{0_i}^T$ and $D_i = \mathbf{I}_{0_i}\mathbf{I}_{0_i}^T$. Reshaping the $N \times NT$ matrix $(G_1\ G_2\ \cdots)$ into an $N \times N \times T$ matrix, we obtain the updated $\mathbf{G}$.

3.2.4 PCA Step. In general, the operators G_i learned via the EM algorithm are not constrained to be orthogonal to each other. As a result, EM may recover matrices that are, for example, linear combinations of the true operators. To recover orthogonal operators (e.g., affine operators), we apply principal component analysis (PCA)[2] to the space of N^2-dimensional vectors $\mathbf{s}_i = \sum_j z_{ji}\mathbf{G}_j = G\mathbf{z}_i$ where $\mathbf{G}_j$ is the N^2-dimensional vectorized form of the $N \times N$ matrix G_j, G is the $N^2 \times T$ matrix whose columns are given by $\mathbf{G}_j$, and $\mathbf{z}_i$ is the T-dimensional transformation vector. Typically, the number of transformation types T is not known. PCA produces an $N^2 \times T'$ matrix G' whose columns are T' orthogonal vectors $\mathbf{G}'_i$ or, equivalently, T' orthogonal operator matrices G'_i of size $N \times N$. These operator matrices obtained from PCA are then used in the E-step for the next iteration of the EM algorithm.

3.2.5 Summary of EM algorithm for Learning Lie Transformation Operators. Repeat until convergence:

- E-step:
  Compute expectations $\langle \mathbf{z}_i \rangle$, $\langle \mathbf{z}_i \mathbf{z}_i^T \rangle$ according to equations 3.12 and 3.13.
- M-step:
  Compute $\mathbf{G}$ and Σ using Equations 3.16 and 3.17.
  Compute orthogonal operators $\mathbf{G}'$ from $\mathbf{G}$ using PCA.
  Use $\mathbf{G}'$ in the next E-step.

4 Results

The learning algorithms described above were tested in three ways. First, we used a synthetic image data set containing up to six different types of affine transformations. To generate synthetic data containing transformations, we used the periodic sinc interpolation function to continuously transform a reference image $\mathbf{I}$ by infinitesimal (subpixel) amounts with no assumptions about the camera sampling mechanism (Marks, 1991). The set of affine operators was derived analytically through a convolution of the numerical differential operators and the interpolation function. Figure 1 shows these analytically derived operators in the form of operator matrices G_i, along with examples of transformations obtained by using them

[2] An interesting alternative worth pursuing is to use independent component analysis (ICA) (Bell & Sejnowski, 1995) instead of PCA, especially when the operators are not known to be orthogonal.

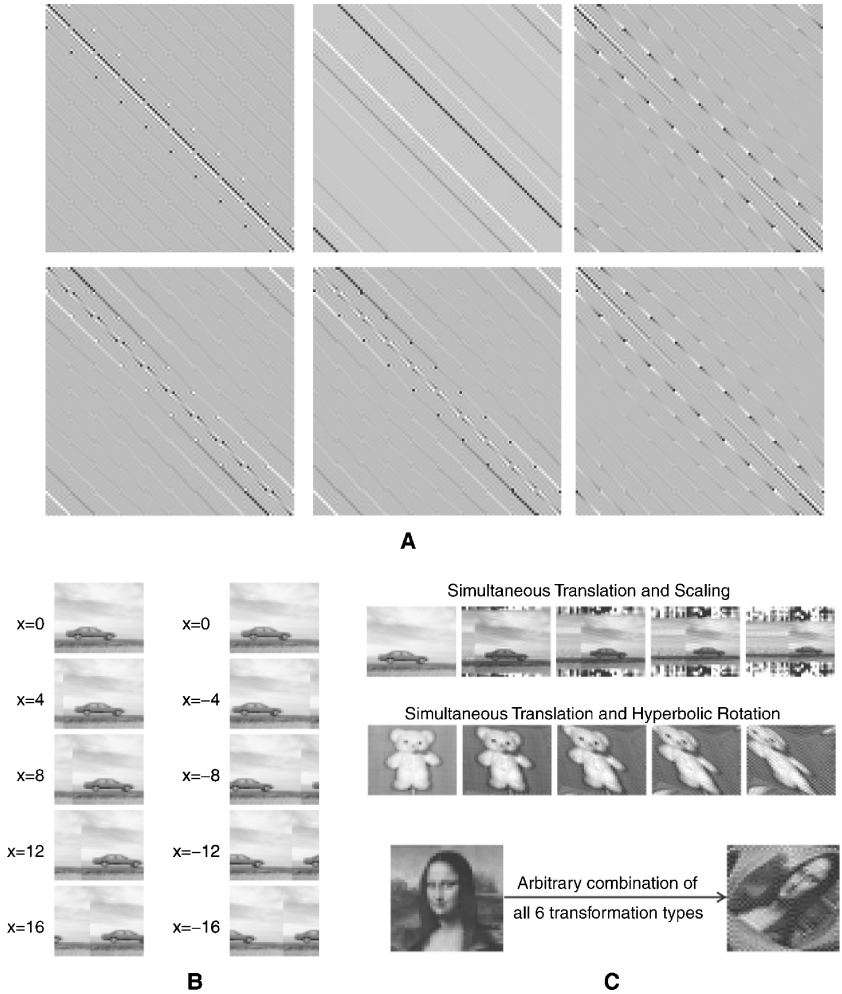

Figure 1: Analytically derived Lie operators and example transformations. (A) The six affine transformation operator matrices for image size 10×10. From left to right and top to bottom, they are shown in the order: horizontal translation, vertical translation, rotation, scaling, parallel hyperbolic deformation, and diagonal hyperbolic deformation. (B) Examples of using the exponential generative model with the horizontal translation analytical operator to translate a gray-scale image of an automobile. (C) Examples illustrating how multiple transformation types can be simultaneously applied to an image using equation 3.1. For the Mona Lisa example, the parameters used were: 3.9 (X-translation), -5.2 (Y-translation), 45 degrees (rotation), 0.1 (scaling), 0.05 (hyperbolic along X/Y), and 0.25 (hyperbolic along diagonal).

in equation 3.1. The analytically derived operators allowed us to test the performance of the gradient-descent- and EM-based learning algorithms as described in section 4.1. Second, we applied the EM-based method to the task of learning unknown transformation operators from natural images. Results from this experiment are reported in section 4.2. A final set of experiments, discussed in section 4.3, focused on using the learned operators for estimating transformations in images.

4.1 Learning Affine Transformations. Previous work (Rao & Ruderman, 1999) demonstrated the utility of using gradient descent for semi-supervised learning of a single Lie group operator: the transformation value $\mathbf{z}$ was assumed to be known for each training image pair. However, in most practical problems, both the transformation values and the operators themselves are unknown. We focus here on this unsupervised learning problem and test the performance of the unsupervised gradient descent approach as well as the EM-based approach described above. We used an artificially generated data set containing one or more of the six types of 2D affine transformations: horizontal translation (H), vertical translation (V), rotation (R), scaling (S), parallel hyperbolic deformation (P), and diagonal hyperbolic deformation (D). We interpret the corresponding transformation parameter vector $\mathbf{z}$ as $[H, V, R, S, P, D]^T$. The synthetic image data set contained 20,000 image pairs. The first image in each pair was generated with uniformly random pixel intensities; the second was obtained by transforming the first image using transformation values z_i picked from zero-mean gaussian distributions with subpixel variance. To simulate the noisy image generation process in equation 3.1, zero-mean gaussian distributed noise with variance of 0.01 was added to each artificially transformed image. An example image pair in this data set is shown in Figure 2. All intensity values were in the range 0 to 1.

4.1.1 Gradient Descent-Based Method. To test the gradient-descent approach in the unsupervised scenario where both the transformation values $\mathbf{z}$ and the operators G_i are unknown, we used the algorithm suggested in section 3.1: alternate between estimating $\mathbf{z}$ using equation 3.7 for each image pair and adapting G_i using equation 3.5 based on the estimated value of $\mathbf{z}$. The learning rate α was initialized to 0.4 and was decreased by dividing it with 1.0001 after each iteration (=100 training pairs). The operator matrices G_i were updated after every 100 pairs of images based on the average value of the gradient.

Figure 3 shows the results of attempting to learn the six affine operator matrices from the training data set using unsupervised gradient descent. As can be seen, the approach fails to learn any of the operator matrices. In fact, the algorithm fails to converge to a stable solution, as shown in Figure 3 (right panel), which plots the optimization value (negative of the

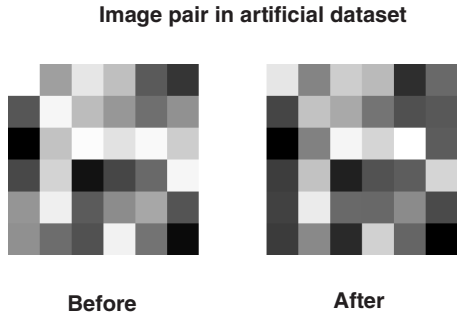

Figure 2: Example of a training image pair from the synthetic data set. The first image is a randomly generated image of size 6×6 pixels. The second image was generated by transforming the first using the following value for $\mathbf{z}$ with components picked from zero-mean gaussian distributions: $[0.0947, 0.0348, -0.0293, 0.0875, -0.0693, -0.0374]^T$.

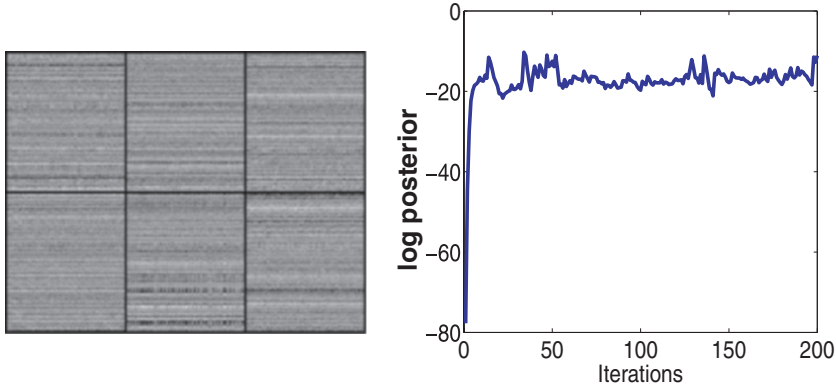

Figure 3: Results from the unsupervised gradient descent algorithm. (Left) Six putative operator matrices learned simultaneously by the unsupervised gradient descent algorithm. It is clear that none of the operators resembles any of the six analytical operators for affine transformations. (Right) The value of the log posterior function (negative of first term only in equation 3.4) over successive iterations. Each iteration involved 100 image pairs. The algorithm failed to converge to a stable set of parameters even after 20,000 training image pairs.

first term in equation 3.4) over 200 iterations (20,000 image pairs). This lack of convergence motivates the use of the EM algorithm.

4.1.2 EM-Based Learning. We first tested the EM algorithm without the PCA step. As shown in Figure 4, the algorithm was unable to recover the six operators, although it did converge to a solution.

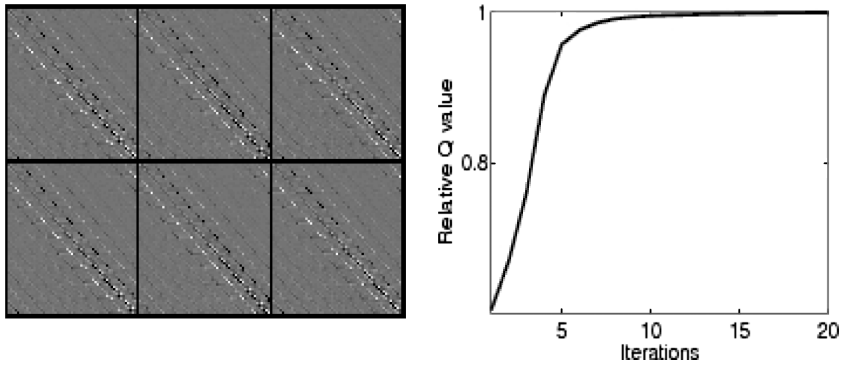

Figure 4: Results from EM without PCA step. (Left) The converged values of six operators learned by the EM algorithm without the intermediate PCA step. The algorithm failed to discover six different operators even though the training data set contained six different types of affine transformations. (Right) The algorithm converged quickly.

We then included the intermediate PCA step (see section 3.2.4) within the EM algorithm to encourage the learning of mutually orthogonal operators. Figure 5 shows that the modified EM algorithm again converges rapidly, but this time to the correct solution (see Figure 5; cf. Figure 1A). To quantify the accuracy of the learning algorithm, we computed the percentage square root error between the learned and analytical operators for both the gradient-descent- and EM-based unsupervised learning algorithms. Figure 6 shows that while the gradient descent algorithm produces operators with significant errors, the percentage error for the EM-based learned operators is consistently below 1.5%.

In a separate set of experiments, we tested the performance of the EM-based algorithm in learning each operator separately rather than simultaneously. The algorithm successfully recovered each operator and in addition converged much faster to the correct solution because no PCA step was necessary in this case.

We also tested the accuracy of the matrices learned by the EM-based algorithm using equation 2.2 to generate images with different transformation values starting from a given reference image. Figure 7 illustrates an experiment where the performance of the learned rotational operator was compared to the analytical operator. The percentage error between the transformed images using the learned versus analytical operator are shown in the bottom panel. The results from all such experiments are summarized in Table 1. The percentage errors are small (<8%) for transformations such as multipixel translations, rotations from 1 to 20 degrees, and shrinkage by

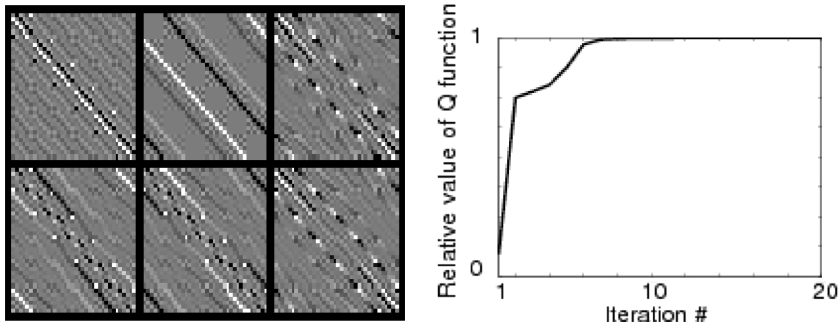

Figure 5: Results from EM algorithm with PCA step. (Left) The six operators learned using the EM algorithm with PCA step. They closely match the analytical operators depicted in Figure 1A. (Right) The value of Q function (normalized to the final attained value) over successive iterations of the E- and M-steps.

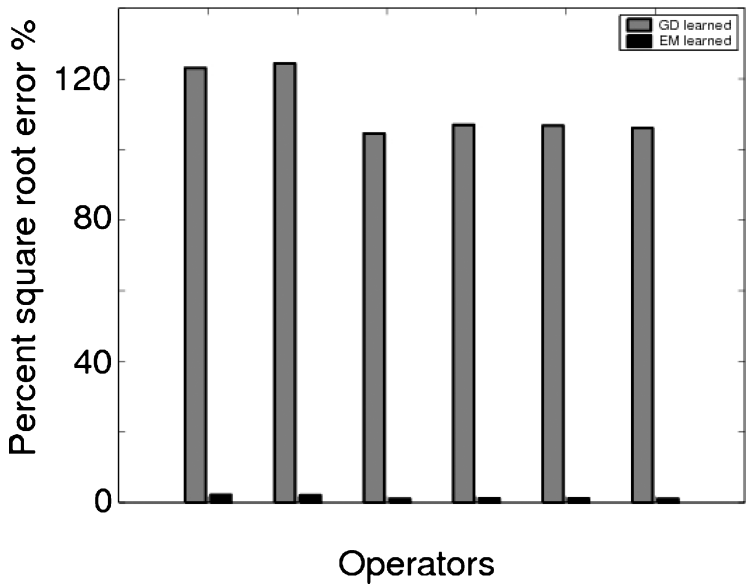

Figure 6: Comparison of operators learned using gradient descent and EM algorithm with PCA step to analytical operators. The first set of bars shows the percentage square root error between operators learned via unsupervised gradient descent and analytical operators for affine transformations. The second set shows the same error for operators learned using EM with PCA. The bars are plotted in the order of transformations H, V, R, S, P, and D (see the text for details).

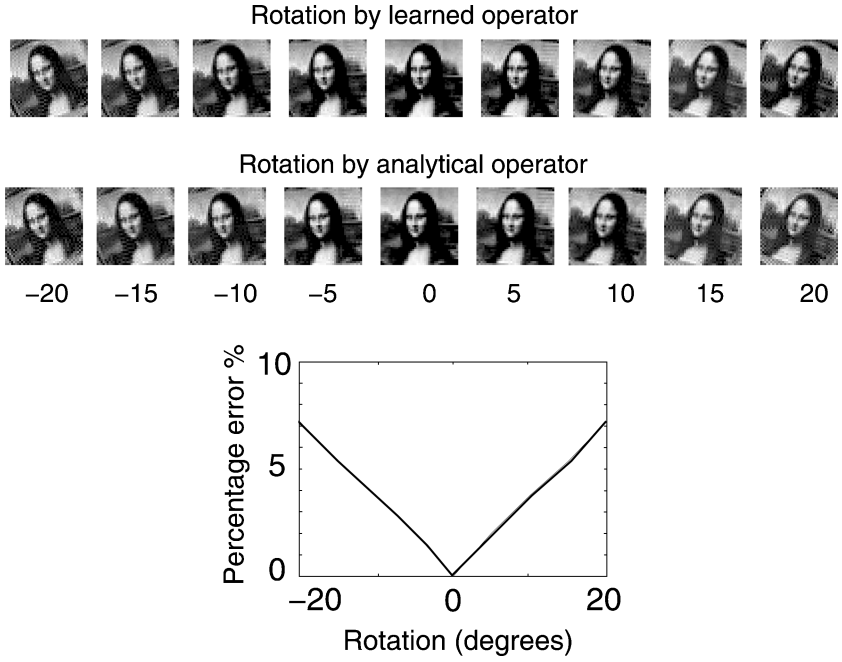

Figure 7: Rotating 2D images using analytical and learned Lie operators. (A) A reference image (labeled 0) of size 40×40 pixels was rotated in the range -20° to $+20^\circ$ using both the analytically derived Lie operator matrix (bottom panel) and the learned matrix (top panel). (B) shows the percentage squared error between images generated using the analytical matrix and the learned matrix, plotted as a function of rotation value.

Table 1: Errors Between the Transformed Images Using the Learned versus Analytical Operators.

Transformation Type	Transformation Value Range	Percentage Error Range
Translation (horizontal	<2 pixels	<0.05%
and vertical) rotation	<20 degrees	<8%
Scaling (shrinkage only)[a]	1-0.67 times	<3%
Hyperbolic (parallel)[a]	<1 units	<20%

[a] Other cases give poor results due to numerical errors in the matrix exponential routine.

30%. For extremely large transformations, the error does become more noticeable primarily due to numerical errors in the matrix exponential routine.

4.2 Learning Lie Operators from Natural Image Sequences. The results in the previous section suggest that the EM-based learning algorithm

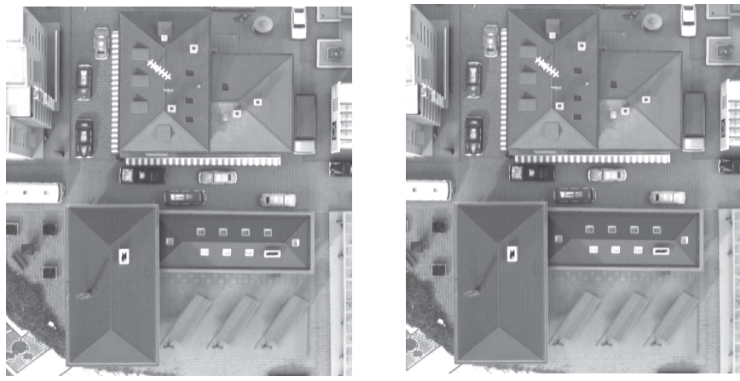

Figure 8: Two images from the Tiny Town natural image sequence. The image sequence consists of 15 frames. The first and last frame are shown. A rightward motion of the camera has caused the scene to shift leftward by a few pixels, as is evident from examining the left and right edges of the two images.

can successfully recover a set of known transformation operators from an artificially constructed data set. To test whether the algorithm can be used to learn unknown operators from natural image sequences, we used the well-known Tiny Town image sequence, consisting of 15 frames of aerial images of a toy town taken from an overhead moving camera. The camera was slowly moving rightward relative to the scene. Figure 8 shows two frames from this sequence. Each image is 480×512 pixels. Consecutive frames in the image sequence exhibit a close-to-infinitesimal leftward translation, making it suitable for testing the EM-based learning algorithm (see equations 3.3 and 3.9). We generated two 10,000 image pair data sets from this image sequence—one for learning a 1D operator and the other for learning a 2D operator. To create each data set, we first randomly selected a pair of consecutive images in the sequence and then chose either a 6×1 (1D case) or a 6×6 patch (2D case) from a random location in each image. To give the learning algorithm examples of both rightward and leftward shifts, we reversed the order of the image patches with 0.5 probability.

Figure 9 shows the 1D and 2D Lie operators learned by the EM algorithm for the natural image sequence. Comparing with translational operators, it is clear that the learned operators resemble the horizontal translation operators except for the assumption of periodicity in the input signal. Indeed, if one derives the operator for horizontal translations using the nonperiodic interpolation function, the learned operator can be seen to be similar (but not identical) to the nonperiodic analytical Lie operator for translation (see Figure 9).

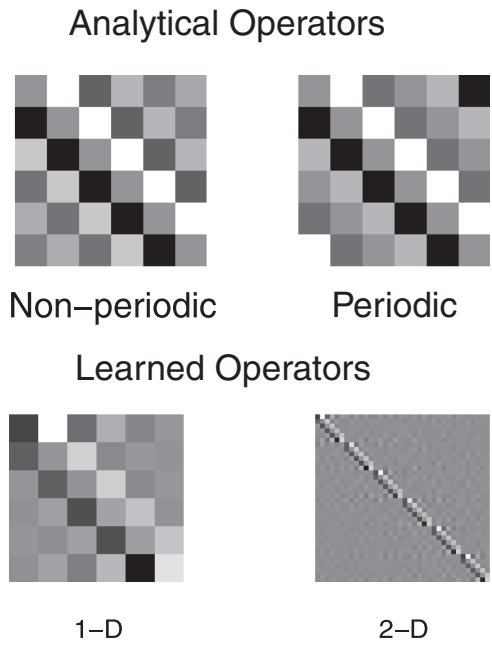

Figure 9: Operators learned from natural images. The top row shows the analytical operator for translation in 1D images based on a nonperiodic (left) and periodic (right) interpolation function. The 1D and 2D operators learned from the natural image sequence are shown below.

4.3 Estimating Transformations Using the Learned Operators. Once a set of transformation operators has been learned, they can be used to estimate and factor out the transformations in images, thereby facilitating visual invariance. Given a pair of input images containing one or more transformations, the transformation values z_i can be estimated using the equations derived in section 3. We examined the efficacy of the learned operators under two conditions: (1) when the image pair contains a single type of transformation and (2) when the image pair contains multiple types of transformations simultaneously. For each case, we investigated two regimes: (a) the subpixel regime, wherein the first-order approximation can be expected to hold true, and (b) the large transformation regime, wherein the full exponential model is required.

4.3.1 Estimating Single Transformations.
Subpixel transformations. We compared the performance of the gradient descent method (see equation 3.6) and the direct method (see equation 3.7) for estimating subpixel transformations in pairs of input images. Figure 10

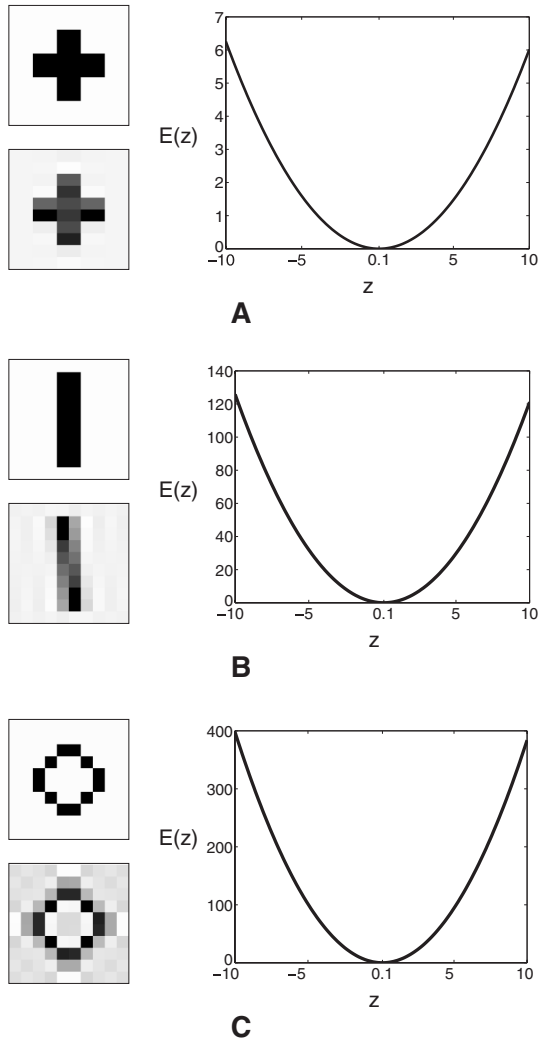

Figure 10: Examples of subpixel transformations and the corresponding optimization surfaces. (A) An original image was translated vertically by 0.1 pixels (pair of images on the left). The plot shows the error function defined by the first term in equation 3.4 for this pair images. (B) The pair images on the left illustrate a rotation of 0.1 radians. The plot on the right shows the corresponding error surface. (C) The pair images on the left depict a scaling of 0.1 (i.e., 1.1 times the original image). The corresponding error surface is shown on the right. Note that for all three subpixel transformations, the error surface contains a unique global minimum.

Table 2: Estimating Subpixel Transformations.

Transformation Type	Actual Value	Analytical Estimate		Learned Estimate	
		Gradient Descent	Direct	Gradient Descent	Direct
Translation	0.1	0.098	0.1	0.098	0.098
Rotation	0.1	0.099	0.1	0.098	0.099
Scaling	0.1	0.1001	0.1001	0.089	0.09

Note: The table compares actual transformation values with estimates obtained using the analytically derived and the learned Lie operator matrices for the image transformations in Figure 10.

shows three pairs of images and the optimization surfaces determined by equation 3.4 as a function of the transformation parameter z. For simplicity, uniform priors were used for all the parameters, allowing the optimization surface to be viewed as an "error surface" reflecting only the first term in equation 3.4. All three surfaces have a single global minimum ($z = 0.1$), which was found by both the gradient descent method and the direct (matrix pseudoinverse) method. Table 2 compares actual transformation values with those estimated using the direct method and the gradient descent method for the transformations depicted in Figure 10. Both the direct method and the gradient descent method recover values close to the actual value used to generate the pair of input images.

Large transformations. An advantage of the Lie approach is that the learned operator matrix can be used to estimate not just subpixel transformations but also larger translations using gradient descent based on the matrix exponential generative model (see equation 3.8). As expected, the optimization function in this case often contains multiple local minima. Figure 11 shows examples of optimization surfaces for three pairs of images containing relatively large transformations (downward translation of 2 pixels in Figure 11A, clockwise rotation of 1 radian in Figure 11B, and scaling by 2 in Figure 11C). In all these cases, the optimization function contains a global minimum and one or more (typically shallower) local minima representing alternate (often cyclic) solutions. The two images on the bottom left of each panel in Figures 11A to 11C show the transformed images corresponding to these minima in the optimization surfaces. Except for Figure 11A (where the second minimum is a cyclic solution), the local minima are shallower and generate transformed images with greater distortion.

To find the global minimum, we performed gradient descent with several equally spaced starting values centered near zero. We picked the transformation estimate that yielded the best (smallest) value for the optimization function. Table 3 compares actual transformation values with the estimates provided by the gradient descent method for a data set of 2D image pairs containing one of three types of transformations (translation, rotation, or

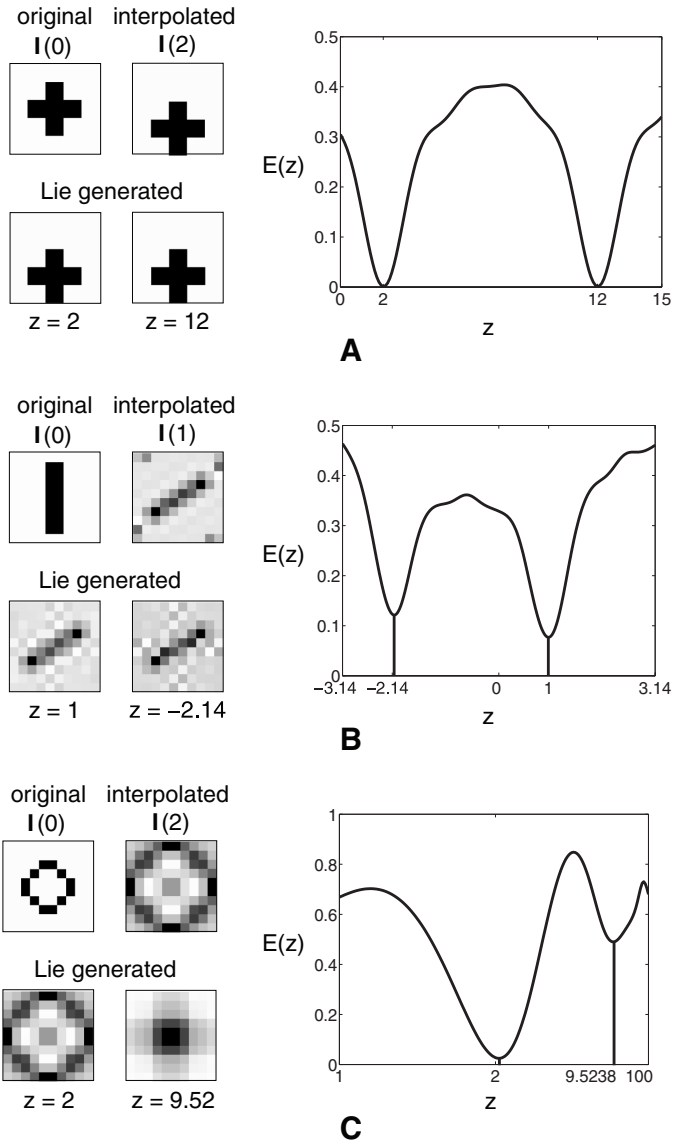

Figure 11: Examples of large transformations and the corresponding optimization surfaces. (A) Example of large translation. The top two images on the left illustrate a downward translation of two pixels obtained by using a periodic sinc interpolation function. The plot on the right shows the error function derived from applying the exponential generative model to these two images. The images representing the two local minima of this function are shown on the left at the bottom. (B, C) Examples of large rotation and large scaling, respectively. The images on the left and the plot on the right follow the scheme in *A*.

Table 3: Estimating Large Transformations in 2D Images.

Transformation Type	Actual Value	Analytical Estimate (by gradient descent)	Learned Estimate (by gradient descent)
Translation	2	2.001	2.002
	3.14	3.140	3.142
Rotation	1	0.998	0.991
	1.41	1.406	1.392
Scaling	2	2.001	2.041
	1.83	1.832	1.850

Note: The table compares actual transformation values with estimates obtained using the analytically derived and the learned Lie operator matrices for an arbitrary set of image transformations.

scaling). Once again, the gradient descent estimates using either the analytical or the learned matrix can be seen to be close to the actual transformation values used to generate the image pair.

4.3.2 Estimating Simultaneous Transformations. Natural images typically contain a mixture of transformations. We tested whether the Lie-based approach could be used to recover multiple simultaneous transformations from image pairs containing all six affine transformations but of different magnitudes. The transformed images were generated by using the periodic interpolation function to sequentially apply each of the six types of transformations. We studied two transformation regimes: (1) simultaneous subpixel transformations and (2) simultaneous large (multipixel) transformations.

Subpixel transformations. We used the matrix pseudoinverse method (see equation 3.7) to recover the six transformation values from pairs of grayscale images that contained simultaneous known subpixel transformations. The performance was found to be comparable to that obtained in the single transformation case.

Large transformations. We tested the performance of the multiple-start gradient descent method based on equation 3.8 for estimating larger (multipixel) simultaneous transformations in image pairs. Figure 12 shows a result from an example image pair. The recovered transformations are approximately correct, though the errors are larger than in the case where the image pairs contain only a single type of transformation (see Table 3). The larger error is partly due to the fact that the estimation method (see equation 3.8) does not take into account the noncommutativity of some sequentially applied transformations (such as translation followed by rotation). An interesting direction of future research is to investigate extensions of the present approach based on, for example, Lie algebras (Helgason, 2001) for more accurately modeling the effects of large, multiple transformations.

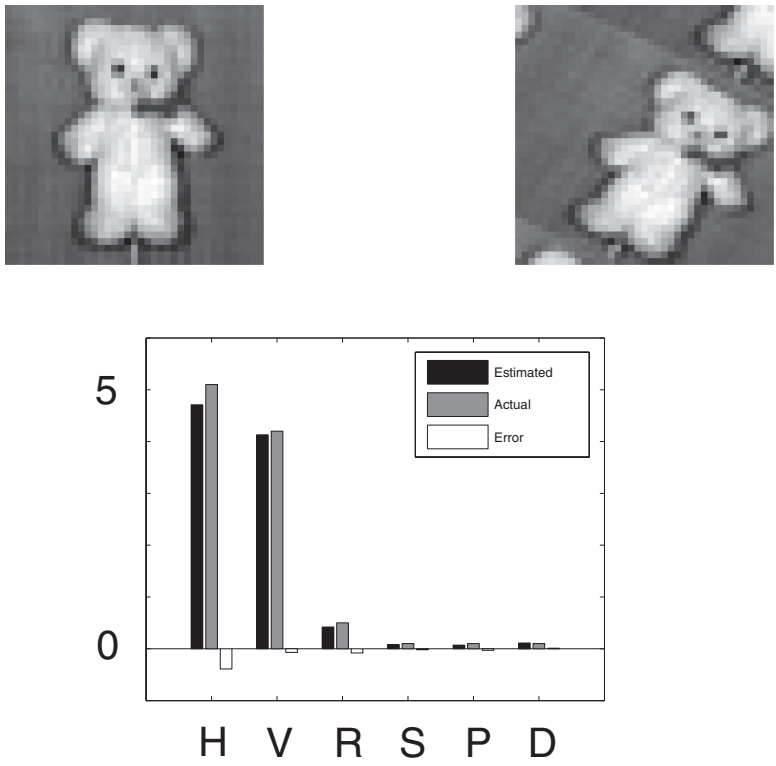

Figure 12: Estimation of large simultaneous transformations. The second image in the top row was obtained by transforming the first using the transformation vector $\mathbf{z} = [5.1, 4.2, 0.5, 0.1, 0.1, 0.1]^T$. The bar plot below shows the estimated transformation value, actual value, and error for each of the six types of affine transformations present in the image pair.

5 Discussion and Conclusion

Our results suggest that it is possible to learn operators (or generators) for Lie transformation groups directly from image data in a completely unsupervised manner. We demonstrated that operators for all six affine image transformations—translations, rotations, scaling, and two types of shear—can be learned directly from image pairs containing examples of these transformations. The learned operators were shown to be almost identical to their analytically derived counterparts. Furthermore, the operators were learned from a training data set containing randomly generated images, indicating that operators can be learned independent of image

content. These operators can be used in conjunction with a matrix exponential-based generative model to transform (or "warp") images by relatively large amounts. Conversely, we showed that the learned operators can be used to estimate one or more transformations directly from input images.

We also provided evidence for the applicability of the approach to natural image sequences by showing that the learning algorithm can recover a novel operator from a well-known image sequence (the Tiny Town sequence) traditionally used to test optic flow algorithms. An interesting research direction therefore is to apply the EM-based algorithm to the task of learning more complex types of transformations directly from video images. For example, nonrigid transformations such as those induced by changes in facial expression or lip movements during speech are hard to model using analytical or physics-based techniques. The unsupervised EM-based algorithm offers an alternate approach based on learning operators for nonrigid transformations directly from examples. Once learned, the operators could potentially be used for estimating the type and amount of nonrigid image transformation in an input video stream.

The Lie generative model (see equation 3.1), together with a generative model for image content (such as in equation 3.2), could be used to address certain challenging problems in computer vision such as simultaneous motion estimation and recognition of objects in natural scenes and image stabilization. The optimization function E in these cases would need to be modified to account for occlusion, multiple motions, and background clutter using, for instance, a layered model (e.g., Adelson & Wang, 1994) or a robust optimization function (as in Black & Jepson, 1996). One could also use sampling-based methods such as particle filtering (Isard & Blake, 1996) to represent multimodal distributions of transformations z_i rather than gaussian distributions. Efforts are currently underway to investigate these extensions.

An important issue concerning the estimation of large transformations based on the exponential generative model is how the globally optimal value can be found efficiently. Several possibilities exist. First, we may impose a prior on the transformation estimate z that favors small values over bigger values; this helps in avoiding solutions that are larger cyclic counterparts of the desired solution, which is closest to zero. This is in fact already implemented in the model with the zero-mean gaussian prior on z and the restriction of search to a region around zero. A second possibility is to use coarse-to-fine techniques, where transformation estimates obtained from images at a coarse scale are used as starting points for estimating transformations at finer scales (see, e.g., Black & Jepson, 1996; Vasconcelos & Lippman, 2005). The coarse-to-fine approach may also help in alleviating the problem of noise in the learned matrices. We hope to investigate such strategies in future work.

Appendix: Formal Derivation of EM Algorithm for Learning Lie Operators

A.1 Derivation of Expectations in the E-Step. In the expectation step (E-step), we need to calculate $\langle \mathbf{z}_i \rangle$ and $\langle \mathbf{z}_i \mathbf{z}_i^T \rangle$. First, define

$$A_i = [\, G_1 \mathbf{I}_{0_i} \quad G_2 \mathbf{I}_{0_i} \quad \cdots \quad G_T \mathbf{I}_{0_i} \,].$$

Then:

$$p(\mathbf{z}_i | \mathbf{I}_i, \mathbf{I}_{0_i}, \mathbf{G}, \Sigma) = \frac{p(\mathbf{I}_i | \mathbf{I}_{0_i}, \mathbf{z}_i, \mathbf{G}, \Sigma) p(\mathbf{z}_i)}{\int p(\mathbf{I}_i | \mathbf{I}_{0_i}, \mathbf{z}_i, \mathbf{G}, \Sigma) p(\mathbf{z}_i) d\mathbf{z}_i} \tag{A.1}$$

$$= \frac{\frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{I}_i - \mathbf{I}_{0_i} - A_i \mathbf{z}_i)^T \Sigma^{-1} (\mathbf{I}_i - \mathbf{I}_{0_i} - A_i \mathbf{z}_i)} \frac{1}{(2\pi)^{\frac{T'}{2}} |\Lambda|^{\frac{1}{2}}} e^{-\frac{1}{2} \mathbf{z}_i^T \Lambda^{-1} \mathbf{z}_i}}{\int p(\mathbf{I}_i | \mathbf{I}_{0_i}, \mathbf{z}_i, \mathbf{G}, \Sigma) p(\mathbf{z}_i) d\mathbf{z}_i}$$

$$= \frac{1}{(2\pi)^{\frac{T'}{2}} |\Sigma_{zi}|^{\frac{1}{2}}} e^{-\frac{1}{2}((\mathbf{z}_i - \mu_i)^T \Sigma_{zi}^{-1} (\mathbf{z}_i - \mu_i))}$$

$$\Sigma_{zi}^{-1} = A_i^T \Sigma^{-1} A_i + \Lambda^{-1} \tag{A.2}$$

$$\mu_i = \Sigma_{zi} A_i^T \Sigma^{-1} (\mathbf{I}_i - \mathbf{I}_{0_i}). \tag{A.3}$$

Here, we assume that the prior distribution of $\mathbf{z}_i$ is a T'-dimensional gaussian distribution, where T' denotes the number of transformations (to distinguish it from matrix transpose T). When $|\Lambda| \to \infty$, the gaussian distribution becomes a uniform distribution.

As $\mathbf{z}_i$ is gaussian, we have:

$$\langle \mathbf{z}_i \rangle = \mu_i \tag{A.4}$$

$$\langle \mathbf{z}_i \mathbf{z}_i^T \rangle = \Sigma_{zi} + \mu_i \mu_i^T. \tag{A.5}$$

A.2 Derivation of the Maximization Step. The expectations calculated in the E-step allow us to maximize Q with respect to $\mathbf{G}$ and Σ. Taking the partial derivatives,

$$\frac{\partial Q}{\partial G_j} = \sum_i \left(- \sum_k \Sigma^{-1} G_k \mathbf{I}_{0_i} \mathbf{I}_{0_i}^T \langle z_{ji} z_{ki} \rangle + \Sigma^{-1} (\mathbf{I}_i - \mathbf{I}_{0_i}) \mathbf{I}_{0_i}^T \langle z_{ji} \rangle \right)$$

$$\frac{\partial Q}{\partial \Sigma^{-1}} = \frac{1}{2} M \Sigma^T - \frac{1}{2} \sum_i (\mathbf{I}_i - \mathbf{I}_{0_i})(\mathbf{I}_i - \mathbf{I}_{0_i})^T$$

$$-\frac{1}{2}\sum_i\sum_j\sum_k G_j \mathbf{I}_{0_i}\mathbf{I}_{0_i}^T G_k^T \langle z_{ji} z_{ki}\rangle$$
$$+\sum_i\sum_j (\mathbf{I}_i - \mathbf{I}_{0_i})\mathbf{I}_{0_i}^T G_j^T \langle z_{ji}\rangle$$

and setting these equal to zero, we obtain:

$$\Sigma = \frac{1}{M}\Bigg(\sum_i (\mathbf{I}_i - \mathbf{I}_{0_i})(\mathbf{I}_i - \mathbf{I}_{0_i})^T + \sum_i\sum_j\sum_k G_j \mathbf{I}_{0_i}\mathbf{I}_{0_i}^T G_k^T \langle z_{ji} z_{ki}\rangle$$
$$-2\sum_i\sum_j (\mathbf{I}_i - \mathbf{I}_{0_i})\mathbf{I}_{0_i}^T G_j^T \langle z_{ji}\rangle\Bigg). \tag{A.6}$$

For G_i, we have

$$G_1\sum_i D_i\langle z_{1i} z_{1i}\rangle + G_2\sum_i D_i\langle z_{1i} z_{2i}\rangle + \ldots = \sum_i C_i\langle z_{1i}\rangle$$
$$G_1\sum_i D_i\langle z_{2i} z_{1i}\rangle + G_2\sum_i D_i\langle z_{2i} z_{2i}\rangle + \ldots = \sum_i C_i\langle z_{2i}\rangle$$
$$\ldots$$

where $D_i = \mathbf{I}_{0_i}\mathbf{I}_{0_i}^T$ and $C_i = (\mathbf{I}_i - \mathbf{I}_{0_i})\mathbf{I}_{0_i}^T$.

Solving this equation using the Kronecker product notation[3] $kron$, we have

$$\begin{pmatrix} G_1 & G_2 & \cdots \end{pmatrix} = \sum_i kron\left(\langle \mathbf{z}_i^T\rangle, C_i\right)\left(\sum_i kron\left(\langle \mathbf{z}_i \mathbf{z}_i^T\rangle, D_i\right)\right)^{-1}. \tag{A.7}$$

Finally, reshaping the $N \times NT$ matrix $(G_1\ G_2\ \cdots)$ into an $N \times N \times T$ matrix, we obtain the updated $\mathbf{G}$.

Acknowledgments

We thank the reviewers for their comments and suggestions. We are also indebted to Dan Ruderman for providing the initial impetus for this work

[3] For an $M \times N$ matrix G and a $P \times Q$ matrix H, the Kronecker (or tensor) product $K = kron(G, H)$ is defined as $K_{ab} = G_{ij}H_{kl}$ where $a = P(i-1)+k$ and $b = Q(j-1)+l$ (Eves, 1980).

and for his early collaboration. This research is supported by an NGA NEGI grant, NSF Career grant no. 133592, and fellowships to R.P.N.R. from the Packard and Sloan foundations.

References

Adelson, E., & Wang, J. (1994). Representing moving images with layers. *IEEE Trans. on Image Processing*, *3*, 625–638.

Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, *7*(6), 1129–1159.

Black, M. J., & Jepson, A. D. (1996). Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proc. of the Fourth European Conference on Computer Vision (ECCV)* (pp. 329–342). Berlin: Springer-Verlag.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Series B*, *39*, 1–38.

Dodwell, P. C. (1983). The Lie transformation group model of visual perception. *Perception and Psychophysics*, *34*(1), 1–16.

Eves, H. (1980). *Elementary matrix theory*. New York: Dover.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, *3*(2), 194–200.

Frey, B. J., & Jojic, N. (1999). 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 (CVPR)* (pp. 416–422). Piscataway, NJ: IEEE.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*, 193–202.

Gibson, J. (1966). *The senses considered as perceptual systems*. Boston: Houghton Mifflin.

Grimes, D., & Rao, R. P. N. (2005). Bilinear sparse coding for invariant vision. *Neural Computation*, *17*(1), 47–73.

Helgason, S. (2001). *Differential geometry, Lie groups, and symmetric spaces*. Providence, RI: American Mathematical Society.

Hinton, G. E. (1987). Learning translation invariant recognition in a massively parallel network. In G. Goos & J. Hartmanis (Eds.), *PARLE: Parallel Architectures and Languages Europe* (pp. 1–13). Berlin: Springer-Verlag.

Isard, M., & Blake, A. (1996). Contour tracking by stochastic propogation of conditional density. In *Proc. of the Fourth European Conference on Computer Vision (ECCV)* (pp. 343–356). New York: Springer.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551.

Marks II, R. J. (1991). *Introduction to Shannon sampling and interpolation theory*. New York: Springer-Verlag.

Nordberg, K. (1994). *Signal representation and processing using operator groups* (Tech. Rep. Linköping Studies in Science and Technology, Dissertations No. 366),

Linköping, Sweden: Department of Electrical Engineering, Linköping University.

Olshausen, B. A., Anderson, C. H., & Essen, D. C. V. (1995). A multiscale dynamic routing circuit for forming size- and position-invariant object representations. *Journal of Computational Neuroscience*, *2*, 45–62.

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607–609.

Pitts, W., & McCulloch, W. (1947). How we know universals: The perception of auditory and visual forms. *Bulletin of Mathematical Biophysics*, *9*, 127–147.

Rao, R. P. N., & Ballard, D. H. (1998). Development of localized oriented receptive fields by learning a translation-invariant code for natural images. *Network: Computation in Neural Systems*, *9*(2), 219–234.

Rao, R. P. N., & Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive field effects. *Nature Neuroscience*, *2*(1), 79–87.

Rao, R. P. N., & Ruderman, D. L. (1999). Learning Lie groups for invariant visual perception. In M. S. Kearns, S. Sollo, & D. Cohn (Eds.), *Advances in neural information processing systems, 11* (pp. 810–816). Cambridge, MA: MIT Press.

Shi, J., & Tomasi, C. (1994). Good features to track. In *Proceedings of IEEE CVPR'94* (pp. 593–600). Piscataway, NJ: IEEE.

Simard, P., LeCun, Y., & Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In C. L. Giles, S. J. Hanson, & J. D. Cowen (Eds.), *Advances in neural information processing systems, 5* (pp. 50–58). San Mateo, CA: Morgan Kaufmann.

Tenenbaum, J. B., & Freeman, W. T. (2000). Separating style and content with bilinear models. *Neural Computation*, *12*(6), 1247–1283.

Van Gool, L., Moons, T., Pauwels, E., & Oosterlinck, A. (1995). Vision and Lie's approach to invariance. *Image and Vision Computing*, *13*(4), 259–277.

Vasconcelos, N., & Lippman, A. (2005). A multiresolution manifold distance for invariant image similarity. *IEEE Transactions on Multimedia*, *7*, 127–142.

Vasilescu, M. A. O., & Terzopoulos, D. (2002). Multilinear Analysis of Image Ensembles: TensorFaces. In *Proc. European Conf. on Computer Vision (ECCV)* (pp. 447–460). Berlin: Springer-Verlag.

Wiskott, L., & Sejnowski, T. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, *14*(4), 715–770.

Received April 7, 2006; accepted December 16, 2006.