Learning the Lie Groups of Visual Invariance

Rajesh P. N. Rao and Xu Miao
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195

E-mail: *rao@cs.washington.edu*, *xm@cs.washington.edu*

Technical Report Number 03-12-01

Dept of Computer Science and Engineering

University of Washington

December 2003

Abstract

A fundamental problem in biological and machine vision is visual invariance: how are objects perceived to be the same despite undergoing transformations such as translations, rotations, and scaling? In this paper, we describe a Bayesian method for learning invariances based on Lie group theory. Previous approaches based on first-order Taylor series expansions of inputs can be regarded as special cases of the Lie group approach, which, in principle, can handle arbitrarily large transformations. Using a matrix-exponential based generative model of images, we derive an unsupervised algorithm for learning Lie group operators from input data containing infinitesimal transformations. Our experimental results show that the Lie operators for translations, rotations, and scaling can be learned directly from training images. We demonstrate that these operators can be used to both generate and estimate transformations in images, thereby providing a basis for achieving visual invariance.

1 Introduction

The recognition of familiar objects in the presence of transformations such as translations, rotations, and scaling is a central problem in perception. The importance of this problem was recognized early by vision researchers such as J. J. Gibson who hypothesized that "constant perception depends on the ability of the individual to detect the invariants" (Gibson, 1966). Pitts and McCulloch were among the first to propose a computational method for perceptual invariance ("knowing universals") (Pitts & McCulloch, 1947). A number of other approaches have since been proposed (Fukushima, 1980; Hinton, 1987; LeCun, Boser, Denker, Henderson, Howard, Hubbard, & Jackel, 1989; Olshausen, Anderson, & Essen, 1995; Tenenbaum & Freeman, 2000; Grimes & Rao, 2003), 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)).

In this paper, we describe a Bayesian method for learning invariances based on the notion of continuous transformations and Lie group theory. We show that previous approaches based on first-order Taylor series expansions of images (Black & Jepson, 1996; Rao & Ballard, 1998) can be regarded as special cases of the Lie group approach. Approaches based on first-order models can account only for 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 which can in principle handle arbitrarily large transformations once the correct transformation operators have been learned. Using Bayesian principles, we derive an on-line

unsupervised algorithm for learning Lie group operators from input data containing infinitesimal transformations. 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. Our experimental results suggest that the proposed method can learn the Lie group operators for 2D translations, rotations, and scaling with a reasonably high degree of accuracy, allowing the use of these learned operators in transformation-invariant vision. Preliminary results from this work were reported in (Rao & Ruderman, 1999).

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 $\mathcal{T}$. We will be interested in the cases where $\mathcal{T}$ is a group i.e. there exists a mapping $f : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$ from pairs of transformations to another transformation such that (a) f is associative, (b) there exists a unique identity transformation, and (c) for every $T \in \mathcal{T}$, there exists a unique inverse transformation of T. These properties seem reasonable to expect in general for transformations on images.

Continuous transformations are those which 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 $F \rightarrow F$. To be concrete, suppose T is parameterized by a

single real number z. Then, the group $\mathcal{T}$ is continuous if the function $T(z) : \Re \rightarrow \mathcal{T}$ is continuous i.e. 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 $z \rightarrow 0$, 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* of 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:

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

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

$$I(z) = e^{zG} I_0 \tag{2}$$

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: $\frac{d}{dz}T = GT$. This suggests an alternate way of deriving Equation 2. Consider the Taylor series expansion of a transformed input $I(z)$ in terms of a previous input $I(0)$:

$$I(z) = I(0) + \frac{dI(0)}{dz}z + \frac{d^2I(0)}{dz^2}\frac{z^2}{2} + \ldots \tag{3}$$

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 3 as: $I(z) = e^{zG}I_0$ which is the same as equation 2 with $I_0 = I(0)$. Thus, some previous approaches based on first-order Taylor series expansions (Black & Jepson, 1996; Rao & Ballard, 1998) can be viewed as special cases of the Lie group-based generative model.

3 Learning Lie Transformation Groups

Our goal is to learn the generators (or operators) G of particular Lie transformation groups directly from input data containing examples of transformations. Note that learning the generator of a transformation effectively allows us to remain invariant to that transformation (see below). We assume 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 pixels 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 "before" image and $\mathbf{I}(z)$ the "after" image containing the infinitesimal transformation. Then, using results from the previous section, we can write the following stochastic generative model for images:

$$\mathbf{I}(z) = e^{zG}\mathbf{I}(0) + \mathbf{n} \tag{4}$$

where $\mathbf{n}$ is assumed to be a zero-mean Gaussian white noise process with variance σ^2. Since learning using this full exponential generative model is difficult due to multiple local minima, we restrict ourselves to transformations that are infinitesimal. The higher order terms then become negligible and we can rewrite the above equation in a more tractable form:

$$\Delta\mathbf{I} = zG\mathbf{I}(0) + \mathbf{n} \tag{5}$$

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

Suppose we are given M image pairs as data. We wish to find the $n \times n$ matrix G and the transformations z which generated the data set. To do so, we take a Bayesian maximum a posteriori approach using Gaussian priors on z and G. The negative log of the posterior probability of generating the data is given by:

$$\begin{aligned} E(G,z) &= -\log P[G,z|\mathbf{I}(z),\mathbf{I}(0)] \\ &= \frac{1}{2\sigma^2}(\Delta\mathbf{I} - zG\mathbf{I}(0))^T(\Delta\mathbf{I} - zG\mathbf{I}(0)) + \frac{1}{2\sigma_z^2}z^2 + \frac{1}{2}\mathbf{g}^T C^{-1}\mathbf{g} \end{aligned} \quad (6)$$

where σ_z^2 is the variance of the zero-mean Gaussian prior on z, $\mathbf{g}$ is the $n^2 \times 1$ vector form of G and C is the covariance matrix associated with the Gaussian prior on G. Extending this equation to multiple image data is accomplished straightforwardly by summing the data-driven term over the image pairs (we assume G is fixed for all images although the transformation z may vary). For the experiments, σ, σ_z and C were chosen to be fixed scalar values but it may be possible to speed up learning and improve accuracy by choosing C based on some knowledge of what we expect for infinitesimal image transformations (for example, we may define each entry in C to be a function only of the distance between pixels associated with the entry and exploit the fact that C needs to be symmetric).

The $n \times n$ generator matrix G can be learned in an unsupervised manner by performing gradient descent on E, thereby maximizing the posterior probability of generating the data:

$$\dot{G} = -\alpha\frac{\partial E}{\partial G} = \alpha(\Delta\mathbf{I} - zG\mathbf{I}(0))(z\mathbf{I}(0))^T - \alpha c(G) \quad (7)$$

where α is a positive constant that governs the learning rate and $c(G)$ is the $n \times n$ matrix form of the $n^2 \times 1$ vector $C^{-1}\mathbf{g}$. The learning rule for G above requires the value of z for the current image pair to be known. We can estimate z by performing gradient

descent on E with respect to z (using a fixed previously learned value for G):

$$\dot{z} \;=\; -\beta\frac{\partial E}{\partial z} = \beta(G\mathbf{I}(0))^T(\Delta\mathbf{I} - zG\mathbf{I}(0)) - \frac{\beta}{\sigma_z^2}z \tag{8}$$

If the prior distribution of z is uniform, we can estimate z directly using the matrix pseudoinverse method:

$$z \;=\; [(G\mathbf{I}(0))^T G\mathbf{I}(0)]^{-1}(G\mathbf{I}(0))^T\Delta\mathbf{I} \tag{9}$$

The learning process thus involves alternating between the fast estimation of z for the given image pair and the slower adaptation of the generator matrix G using this z. Figure 1A depicts a possible network implementation of the proposed approach to invariant vision. The implementation, which is reminiscent of the division of labor between the dorsal and ventral streams in primate visual cortex (Felleman & Van Essen, 1991), uses two parallel but cooperating networks, one estimating object identity and the other estimating object transformations. The object network is based on a standard linear generative model of the form: $\mathbf{I}(0) = U\mathbf{r} + \mathbf{n}_0$ where U is a matrix of learned object "features" and $\mathbf{r}$ is the feature vector for the object in $\mathbf{I}(0)$ (see, for example, (Olshausen & Field, 1996; Rao & Ballard, 1997)). Perceptual constancy is achieved due to the fact that the estimate of object identity remains stable in the first network as the second network attempts to account for any transformations being induced in the image, appropriately conveying the type of transformation being induced in its estimate for z (see (Rao & Ballard, 1998) for more details).

The estimation rule for z given above is based on a first-order model (Equation 5) and is 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

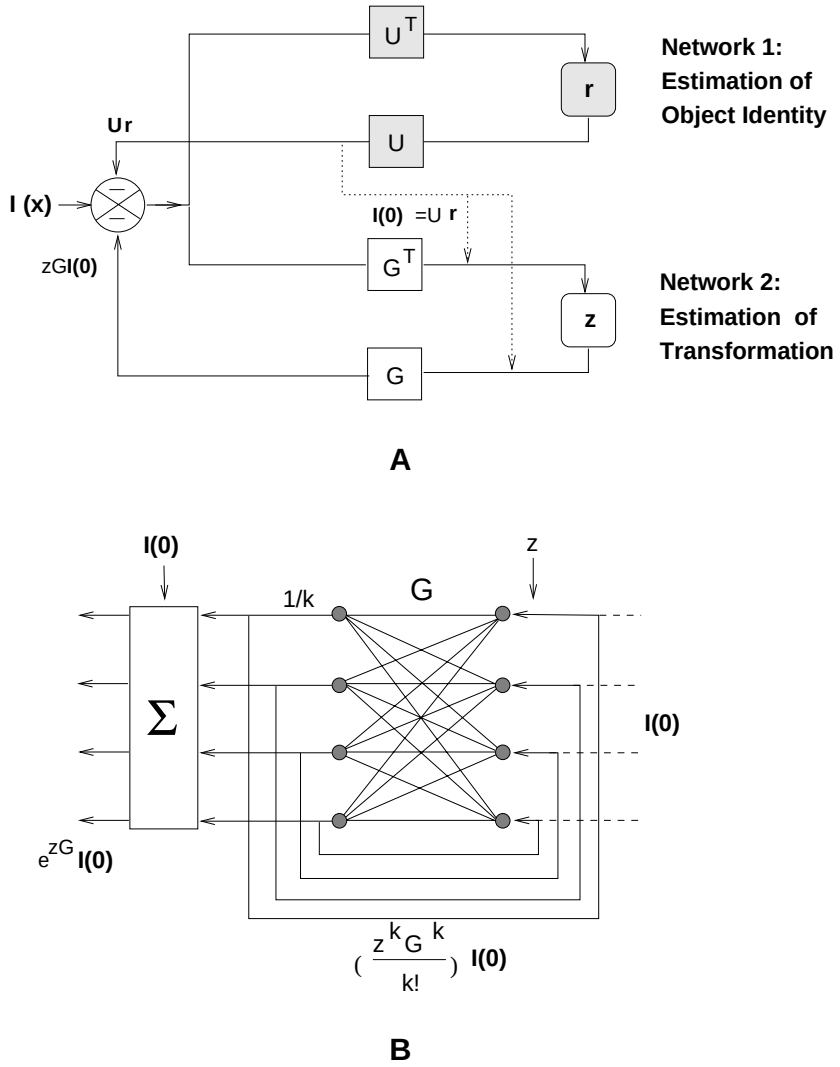

Figure 1: **Network Architecture**. (A) An implementation of the proposed approach to invariant vision involving two cooperating recurrent networks, one estimating transformations and the other estimating object features. The latter supplies the reference image $\mathbf{I}(0)$ to the transformation network based on a set of basis vectors U and their coefficients $\mathbf{r}$ ($\mathbf{I}(0) = U\mathbf{r}$). More details on this approach can be found in (Rao & Ballard, 1998). (B) A locally recurrent elaboration of the transformation network for implementing Equation 10. The network computes $e^{zG}\mathbf{I}(0) = \mathbf{I}(0) + \sum_k (z^k G^k / k!)\mathbf{I}(0)$.

(Equation 4):

$$\dot{z} \;=\; \gamma(e^{zG}G\mathbf{I}(0))^T(\mathbf{I}(z) - e^{zG}\mathbf{I}(0)) - \frac{\gamma}{\sigma_z^2}z \tag{10}$$

Figure 1B shows a locally recurrent network implementation of the matrix exponential computation required by the above equation. This implementation assumes that the dynamics of the network is more rapid than the rate at which the image changes.

4 Results

4.1 Training Paradigm and Interpolation Function

For the purpose of evaluating the algorithm, we generated synthetic training data by subjecting a randomly generated image (containing uniformly random pixel intensities) to a known transformation. Consider a 1-D image $\mathbf{I}$ with N pixels given by $I[j]$, $j = 0, \ldots, N-1$. To be able to continuously transform $\mathbf{I}$ sampled at discrete pixel locations by infinitesimal (sub-pixel) amounts, we need to employ an interpolation function. We make use of the Shannon-Whittaker theorem (Marks II, 1991) stating that any band-limited signal $I[j]$, with j being any real number, is uniquely specified by its sufficiently close equally spaced discrete samples.

If we assume that the signal is periodic i.e. $I[j+N] = I[j]$ for all j, the Shannon-Whittaker theorem in one dimension (Marks II, 1991) states that:

$$I[j] = \sum_{m=0}^{N-1} I[m] \sum_{r=-\infty}^{\infty} \text{sinc}(\pi(j - m - Nr)), \tag{11}$$

where $\text{sinc}(x) = \sin(x)/x$. After some algebraic manipulation and simplification, this can be reduced to:

$$I[j] = \sum_{m=0}^{N-1} I[m]Q(j-m), \tag{12}$$

where the interpolation function Q is given by:

$$Q(x) = (1/N)[1 + 2 \sum_{p=1}^{N/2-1} \cos(2\pi px/N)]. \quad (13)$$

Figure 2A shows this interpolation function for interpolating 1-D images with $N = 20$ pixels. Note that the function shown spans the range from -20 to +20 pixels to allow its use in Equation 12. Note also that the shape of the interpolation function reflects our assumption of a periodic signal. Other interpolation functions which do not assume periodicity may also be used, although they may not offer the same favorable properties as the interpolation function above prescribed by the Shannon-Whittaker theorem (Marks II, 1991).

The interpolation function described above allows one to generate images with arbitrary sub-pixel transformation. For example, to translate a 1-D image $\mathbf{I}$ by an infinitesimal amount $x \in \Re$, we may use:

$$I[j + x] = \sum_{m=0}^{N-1} I[m]Q(j + x - m). \quad (14)$$

Figure 2B illustrates this operation by showing the specific interpolation function Q for determining the pixel value in location $j = 9$ for a 0.5-pixel rightward translation of a 1-D image with $N = 20$ pixels (here, $x = -0.5$). To translate, rotate or scale a 2-D image, we used 2-D variations of the 1-D interpolation equation above.

In addition to enabling images with known transformations to be generated, the interpolation function also allows one to derive an analytical expression for the Lie operator matrix directly from the derivative of Q. This allows us to evaluate the results of learning. For example, the analytical Lie operator for 1-D translation is given by $\frac{d}{dx}$ as applied to the interpolation function $Q(x)$. Figure 3A shows this operator (in the form of the matrix G) for translation of 20-pixel images (bright pixels = positive values,

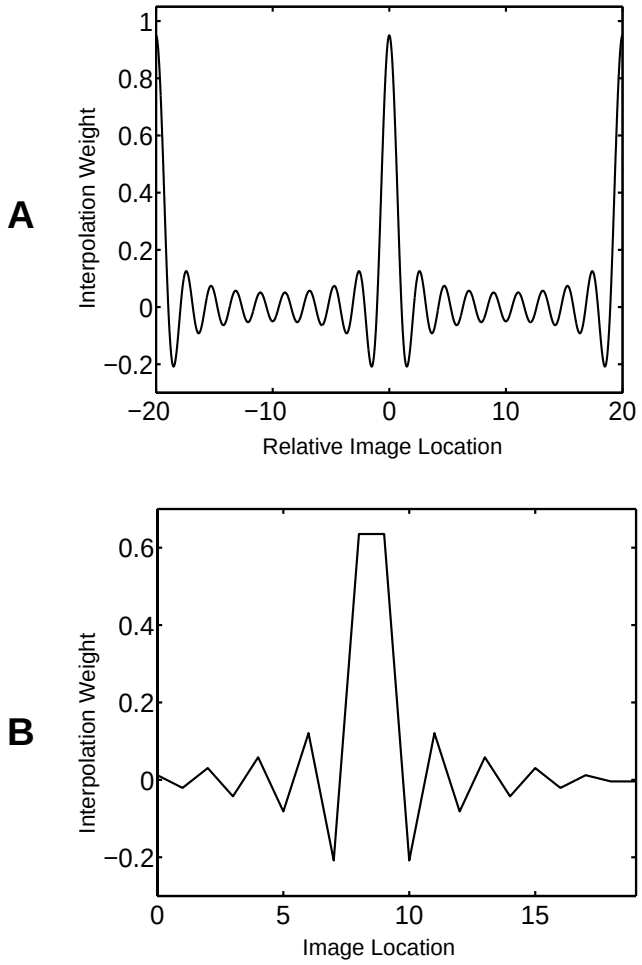

Figure 2: **Interpolation Functions**. (A) The general interpolation function Q used to generate training data (assuming periodic, band-limited signals). (B) Example of specific interpolation function used for determining the pixel value in location 9 for a rightward shift of 0.5 pixels in a 20-pixel 1-D image (see Equation 14).

dark = negative). Also shown alongside is one of the rows of G (row 10) representing the analytical Lie operator centered on pixel location 10.

4.2 Learning 1-D Translations

We used Equation 7 and $50,000$ training image pairs to learn the generator matrix for 1-D translations in 20-pixel images. Training image pairs were obtained by translating a randomly generated first image left or right by 0.5 pixels. We used $C^{-1} = 0.0001$. The learning rate α was initialized to 0.4 and was decreased by dividing it with 1.0001 after each training pair.

Figure 3B shows the results. As expected, the rows of the learned G matrix are identical except for a shift: the same differential operator (shown in Figure 3B) is applied at each image location. The results suggest that the learning algorithm was able to learn a good approximation of the true generator matrix (to within an arbitrary multiplicative scaling factor).

To evaluate the learned matrix G, we translated a given reference image $\mathbf{I}(0)$ using both the analytical and the learned G matrix (Figure 4). We computed the percentage error between pairs of transformed images obtained using the analytical and learned matrices respectively as the total sum of squared pixel-wise errors between the two images divided by the sum of squared pixel values for the analytically-predicted image. As shown in Figure 4C and as quantified in Figure 4D, the learned matrix performs well for translations up to +/-5 pixels but introduces some minor artifacts for larger transformations. We believe this is due to amplification of noise in the learned matrix after applying the matrix exponentiation operation in the generative model (Equation 4). This behavior also occurs for other learned transformation matrices. Possible remedies are discussed in Section 5.

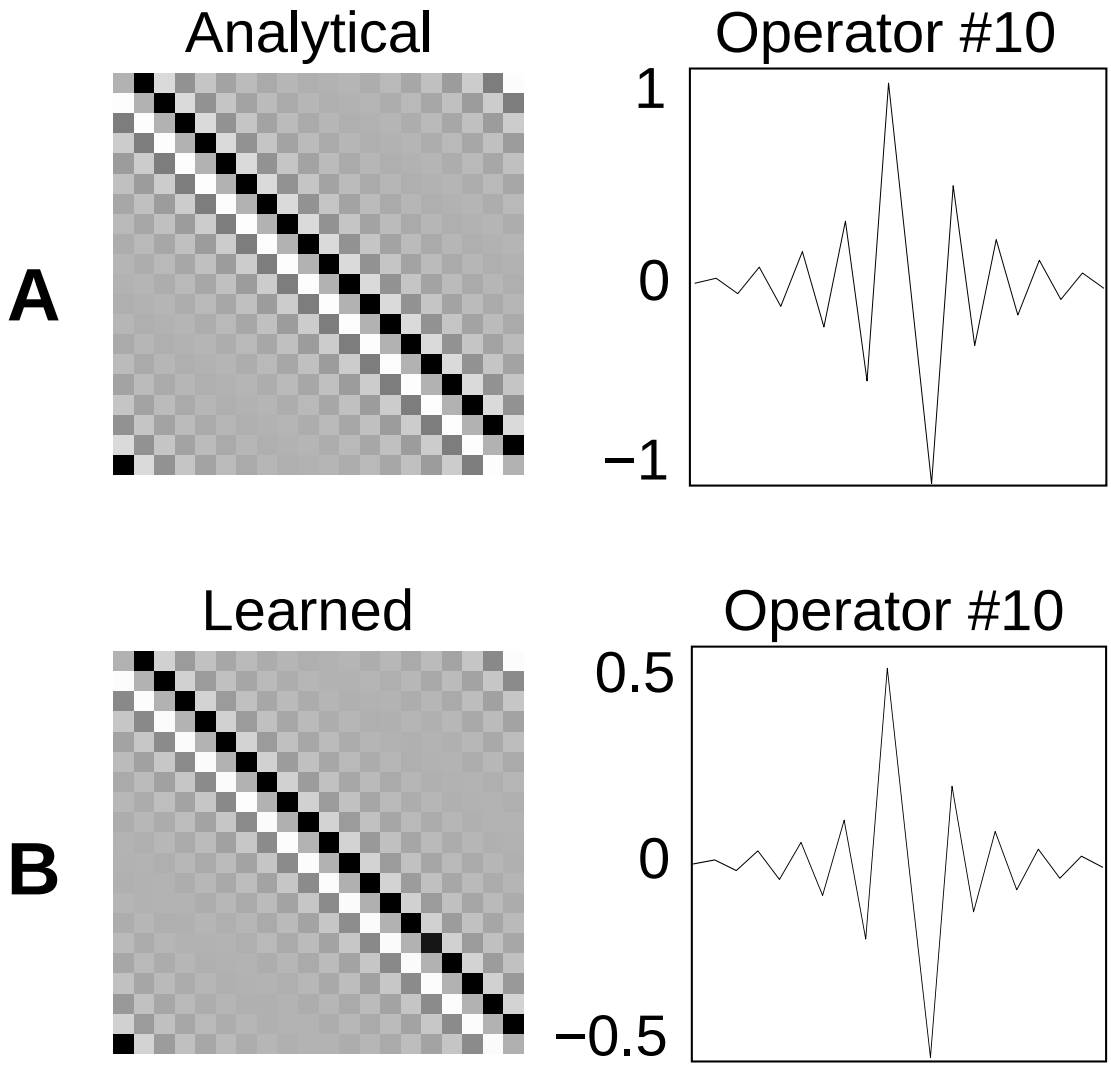

Figure 3: **Learned Lie Operators for 1-D Translations**. (A) Analytically-derived 20×20 Lie operator matrix G and the operator for the 10th pixel (10th row of G). (B) Learned G matrix and a plot of the 10th operator. In the two matrix images, bright pixels denote positive value while dark pixels denote negative values.

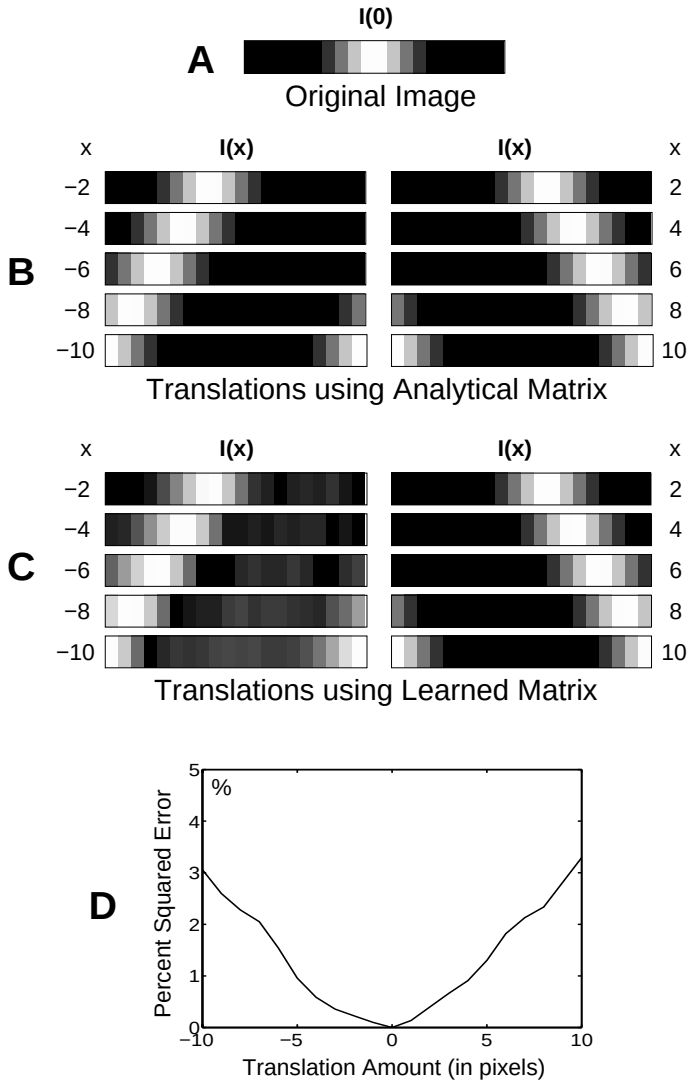

Figure 4: **Translating Images using Analytical and Learned 1-D Operators**. (A) Original reference image $\mathbf{I}(0)$ containing 10 pixels. (B) Images showing the result of translating $\mathbf{I}(0)$ by amounts in the range of -2 to -10 and +2 to +10 pixels. These images were generated using the exponential generative model (Equation 2) with the analytical translation matrix G shown in Figure 3A. (C) Images generated using the learned matrix for translation (Figure 3B) instead of the analytical matrix for the same values of translations as above. (D) Plot showing the percent total squared pixelwise error between the images in (B) and (C) as a function of translation amount.

4.3 Learning 2-D Translations

To test the applicability of the learning algorithm to the 2-D case, we generated image pairs by translating an original random image either horizontally or vertically by 0.5 pixels. The images were of size 10×10 pixels. These training image pairs were used to learn separate Lie operators for translation along the x- and y- axis.

The analytical Lie operators for 2-D translation along the x and y directions are given by $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ respectively applied to the interpolation function $Q(x, y)$. Figure 5 shows that the learned Lie operators for horizontal and vertical translation approximate the corresponding analytically derived operators. To directly compare the analytical with the learned operator, we translated a reference image by varying amounts horizontally and vertically using the x- and y-operators respectively, and diagonally using both the operators. The resulting images are shown in Figures 6A, 6B and 7A. The percentage error between transformed images obtained using the analytical versus learned matrices was computed as in the previous section. The errors were negligible (less than 1%) for horizontal and vertical translations from -2 to +2 pixels and less than 2% for diagonal translations from -3 to +3 pixels (see Figures 6C, 6D, and 7B).

4.4 Learning 2-D Rotations

For learning image plane rotations, training image pairs were generated by infinitesimally rotating 2D images containing random pixel intensities 0.1 radians clockwise or counterclockwise. The Lie operator matrix was then learned from these image pairs.

The analytical operator for rotations is given by $-y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}$ applied to the interpolation function $Q(x, y)$. Figure 8 compares the learned operator matrix with the analytical matrix for rotation. The accuracy of the learned matrix was tested by using it in Equation 2 for various rotations θ. As shown in Figure 9A for the 10×10 case, the

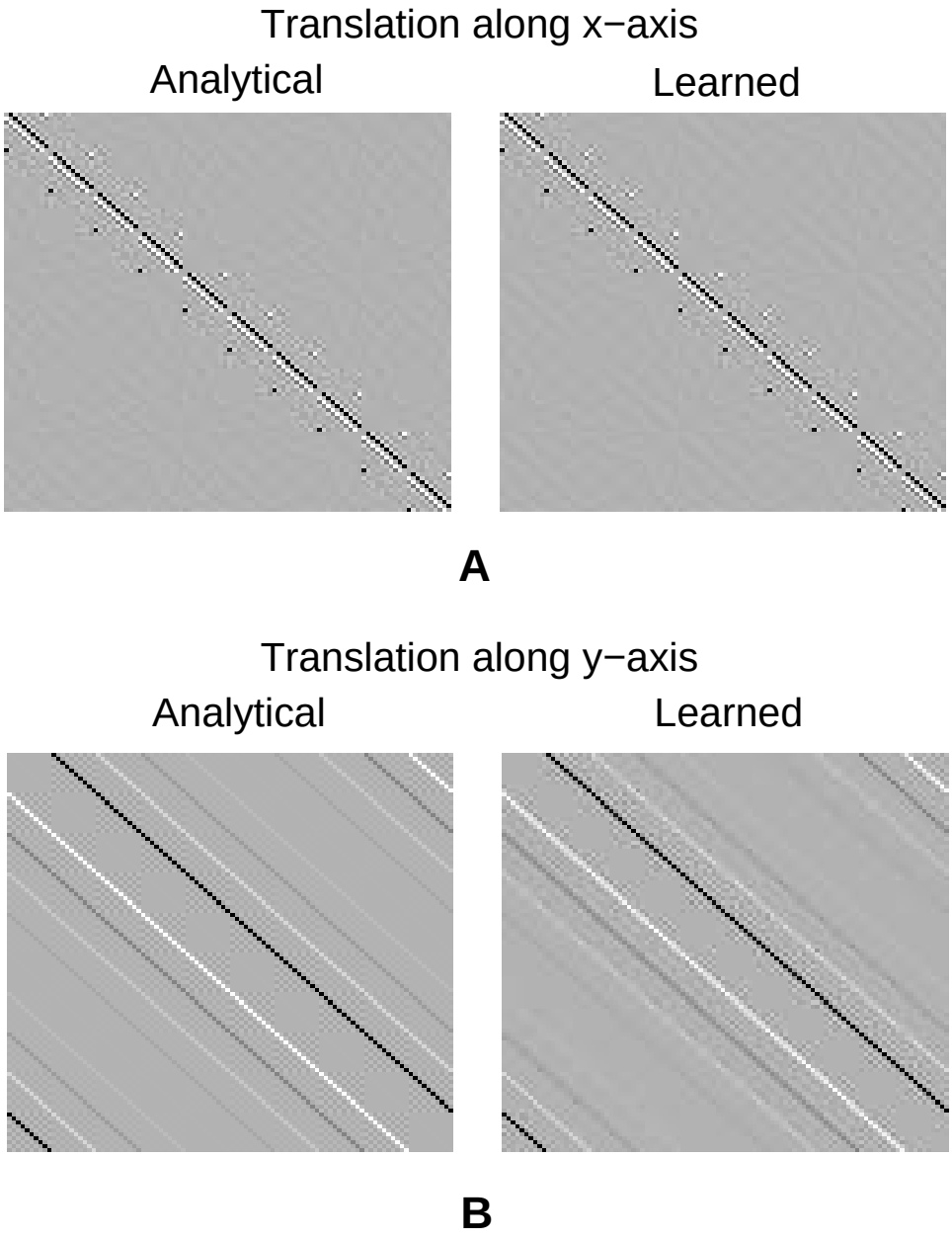

Figure 5: **Learned Lie Operators for 2-D Translations**. (A) Analytical and learned Lie operator matrices for horizontal translation in 10×10 images. (B) Analytical and learned Lie operator matrices for vertical translation in 10×10 images. Bright pixels denote positive values while dark pixels denote negative values.

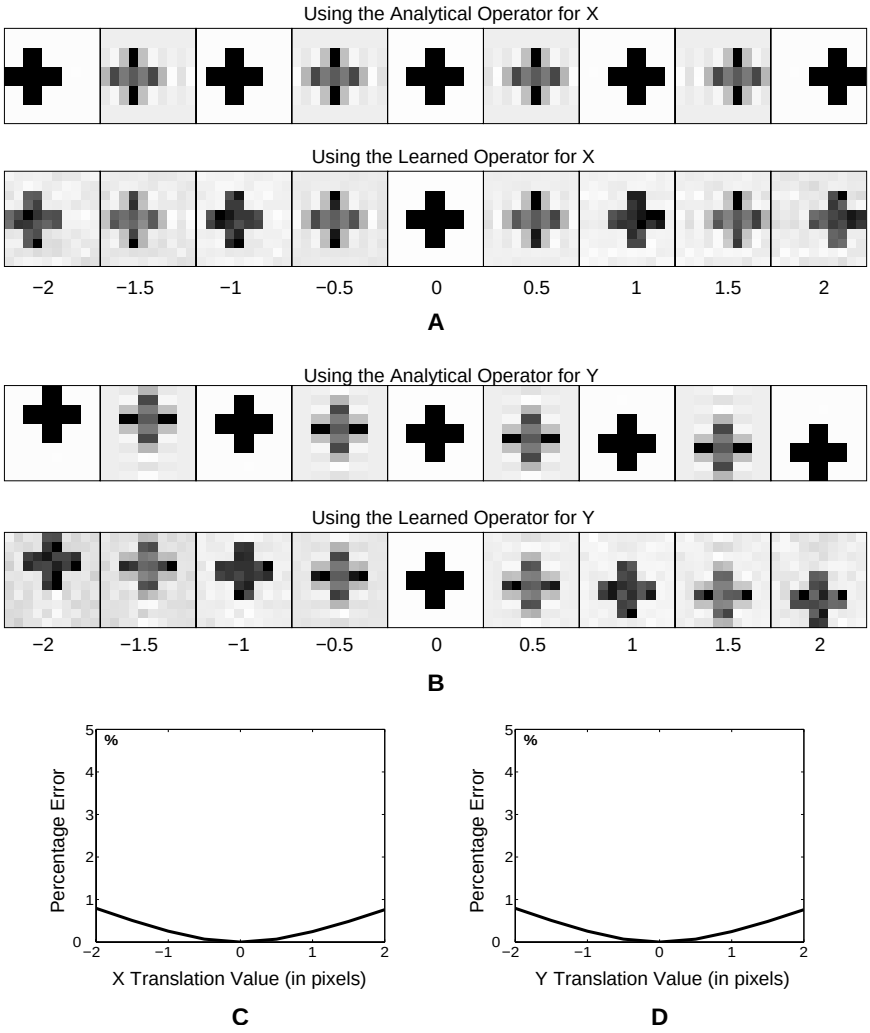

Figure 6: **Translating 2-D Images using Analytical and Learned 2-D Operators**. (A) Comparison of the performance of the analytically-derived and learned 2-D operator for translating images in the X-direction. An original image (labeled 0) was translated horizontally in the range -2 to +2 pixels using Equation 2. (B) Comparison of the performance of the analytically-derived and learned 2-D operators for vertical translation. Results of using Equation 2 are shown for vertical translations in the range -2 to +2 pixels. (C) & (D) Percent total squared pixelwise error between corresponding images generated by using the analytical and learned operators for X- and Y-directions respectively, plotted as a function of translation amount in pixels.

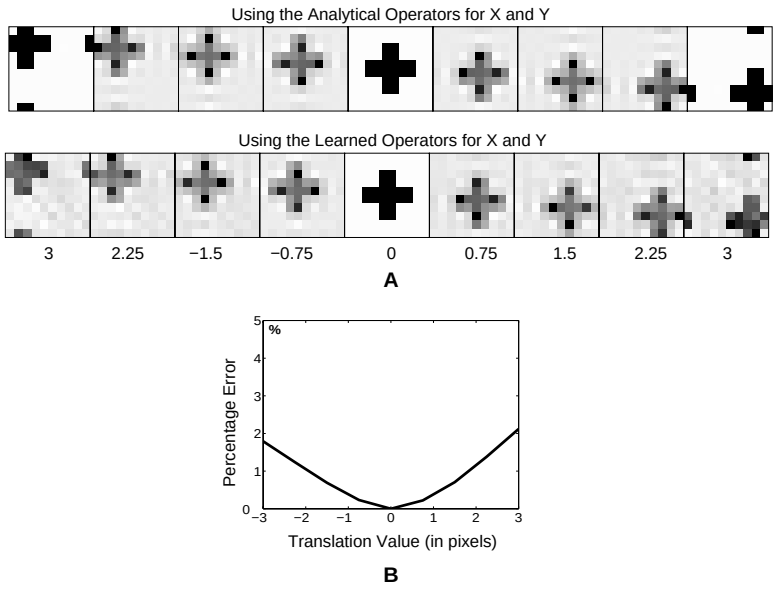

Figure 7: **Diagonal Translation using X and Y Operators Simultaneously**. (A) An original image (labeled 0) was translated diagonally using the X- and Y-translation operators simultaneously in Equation 2. (B) Percent total squared error between corresponding images in (A) plotted as a function of diagonal translation amount.

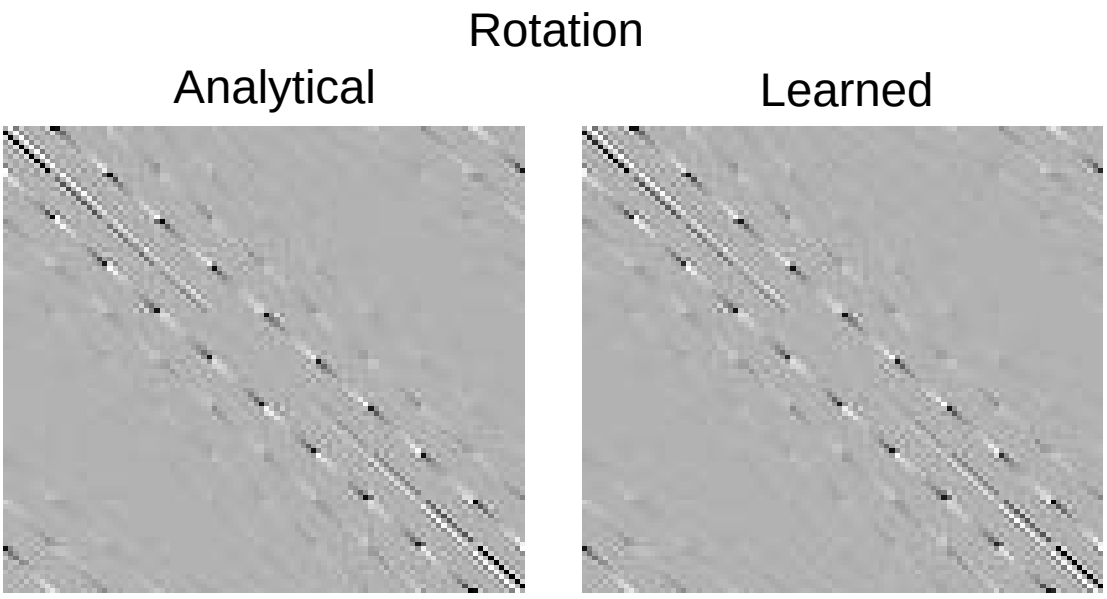

Figure 8: **Lie Operators for Rotation**. The figure shows the analytical and learned Lie rotation matrices for 10×10 images.

learned matrix appears to be able to rotate a given reference image between -90° and $+90^\circ$ about an initial position. For the larger rotations, some minor artifacts begin to appear but the percentage squared error between the transformed images produced by the analytical and learned matrices remained less than 1.5% for the range of rotations studied (Figure 9B).

4.5 Learning 2-D Scaling

The Lie operator for scaling was learned from training image pairs generated by infinitesimally scaling random pixel images using the interpolation function $Q(x, y)$. The analytical operator for scaling is given by $x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y}$ applied to $Q(x, y)$. As seen in Figure 10, the learned operator matrix for scaling approximates the analytically-derived matrix closely. The accuracy of the learned matrix was tested by using it in Equation 2 to generate images with different degrees of scaling starting from a given reference im-

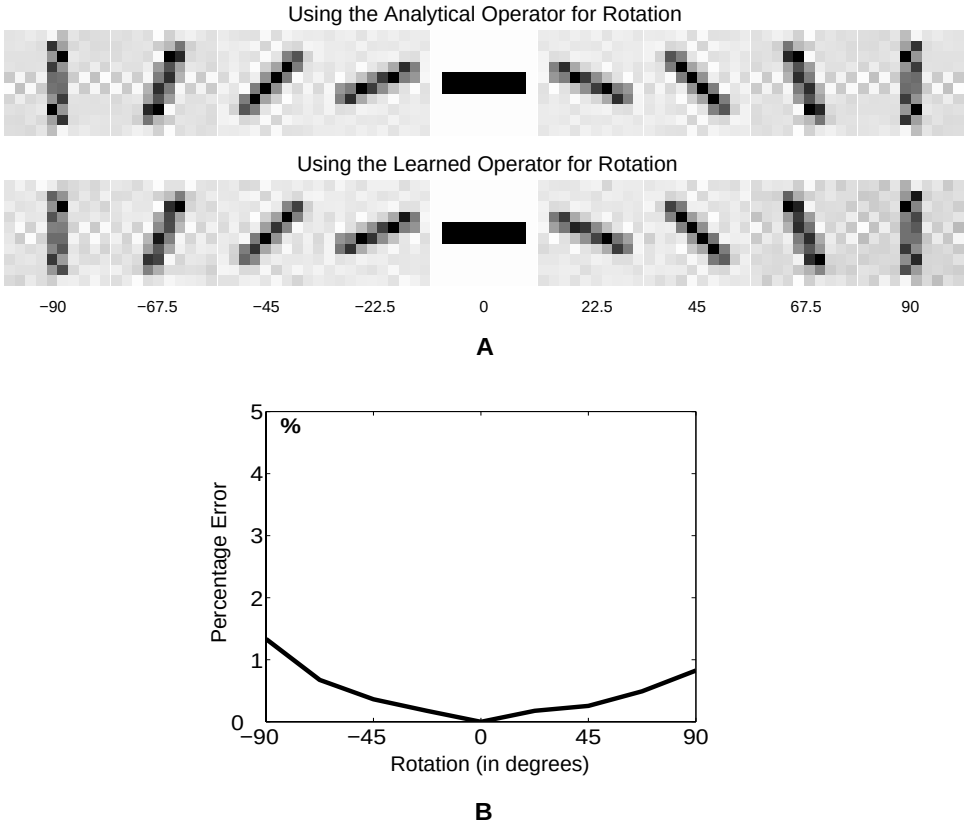

Figure 9: **Rotating 2-D Images using Analytical and Learned 2-D Operators**. (A) A reference image (labeled 0) depicting a dark horizontal bar on a white background was rotated in the range -90° to $+90^\circ$ using both the analytically-derived Lie operator matrix (top panel) and the learned matrix (bottom panel). (B) shows the percent squared error between images generated using the analytical matrix and the learned matrix, plotted as a function of rotation value.

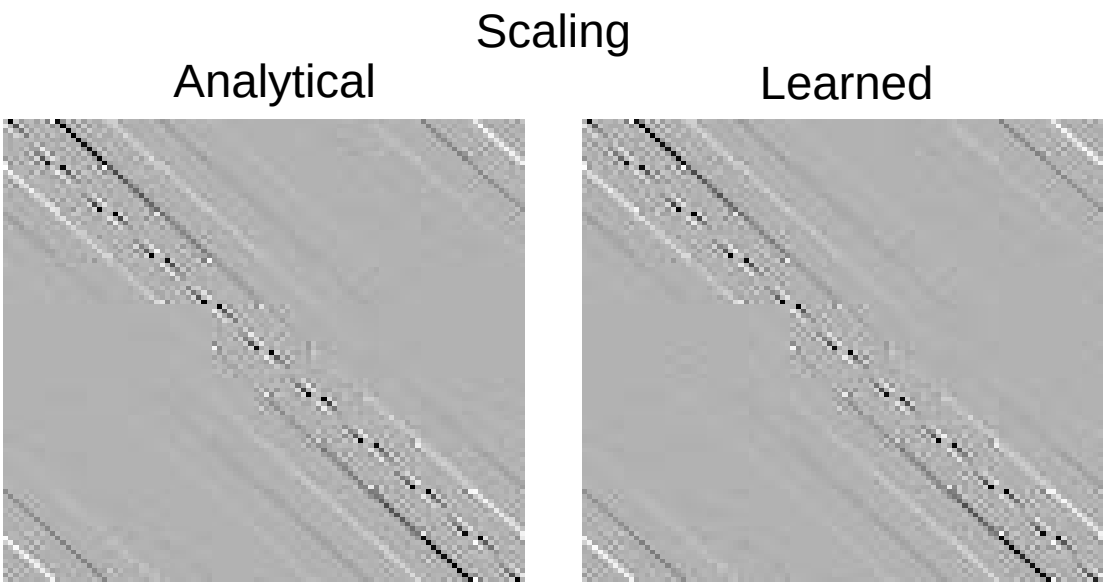

Figure 10: **Lie Operators for Scaling**. The analytical and learned Lie matrices for scaling are shown for 10×10 images.

age. As shown in Figure 11A, the learned matrix performs favorably when compared to the analytical matrix. The percent squared error between corresponding images produced by the analytical and learned matrices remained below 5% for scaling values up to 2 (Figure 11B).

4.6 Estimating Transformations Directly from Image Pairs

Given a pair of input images containing a transformation, the transformation value z can be estimated using the equations derived in Section 3. We examined the efficacy of these equations in two regimes: (a) the sub-pixel 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.

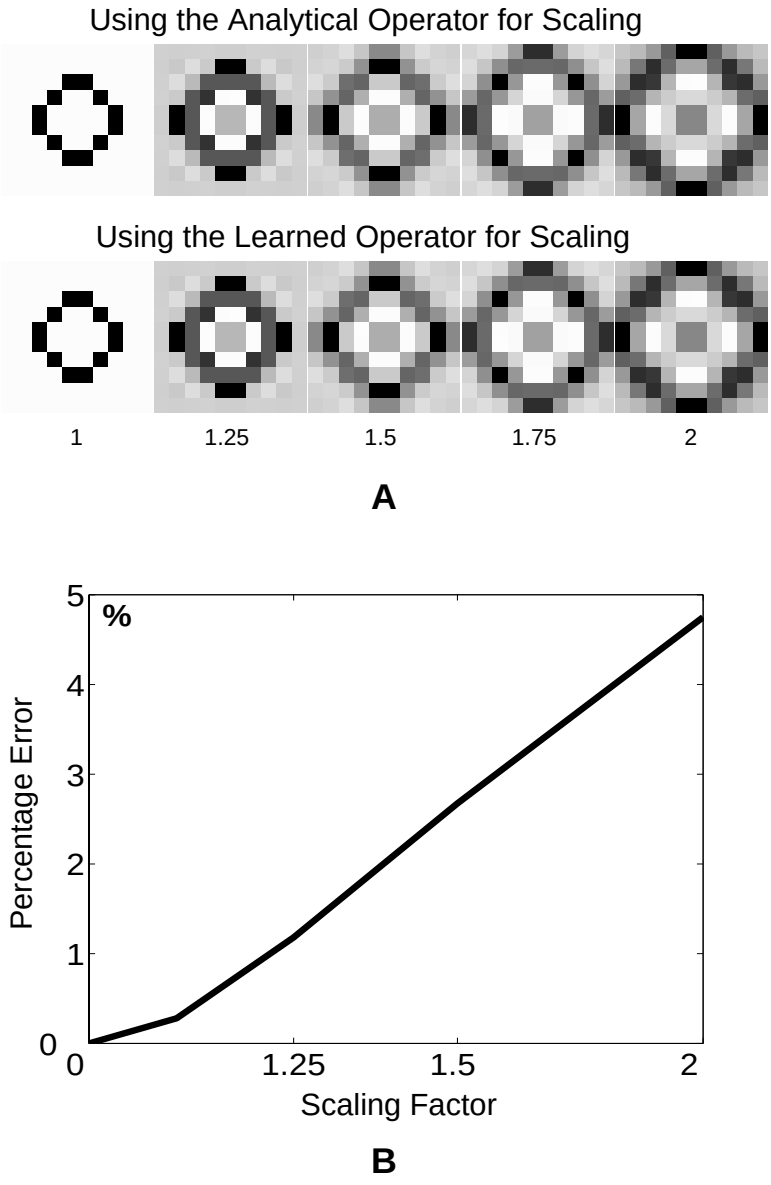

Figure 11: **Scaling in 2-D Images using Analytical and Learned 2-D Operators**. (A) A reference image consisting of a dark circle on a white background was scaled up to twice its original size using both the analytically-derived matrix (top panel) and the learned matrix (bottom panel) for scaling. (B) shows the percent squared error between images generated using the analytical matrix and the learned matrix, plotted as a function of the scaling factor.

4.6.1 Estimating Sub-Pixel Transformations

We compared the performance of the gradient descent method (Equation 8) and the direct method (Equation 9) for estimating subpixel transformations in pairs of input images. Figure 12 shows three pairs of images and the optimization surfaces determined by Equation 6 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 6. 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. The table in Figure 13 compares actual transformation values with those estimated using the direct method and the gradient descent method for the transformations depicted in Figure 12. Both the direct method and the gradient descent method recover values close to the actual value used to the generate the pair of input images.

4.6.2 Estimating Large Transformations

An advantage of the Lie approach is that the the learned generator 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 10). Unfortunately, the optimization function in this case often contains multiple local minima. Figure 14 shows examples of optimization surfaces for three pairs of images containing relatively large transformations (downward translation of 2 pixels in Figure 14A, clockwise rotation of 1 radian in Figure 14B, and scaling by 2 in Figure 14C). 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 Figure 14A, 14B, and 14C show the trans-

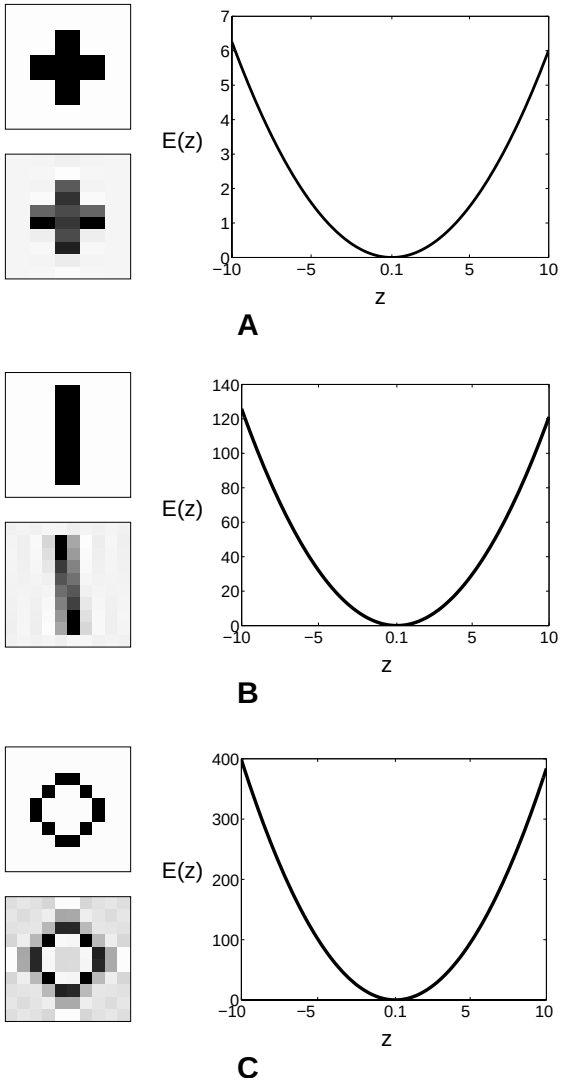

Figure 12: **Examples of Subpixel Transformations and the Corresponding Optimization Surfaces**. (A) An original image was translated vertically by 0.1 pixels (pair of images on the left). The plot shows the error function defined by the first term in Equation 6 for this pair of images. (B) The pair of 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.

Tranformation Type	Actual Value	Analytical Estimate		Learned Estimate	
		Grad. Desc.	Direct	Grad. Desc.	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

Figure 13: **Estimating Subpixel Transformations**. 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 12.

formed images corresponding to these minima in the optimization surfaces. Except for Figure 14A (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 which yielded the best (smallest) value for the optimization function. Figure 15 illustrates the performance of this method for estimating large translations in 1-D images using the learned Lie operator matrix for 1-D translations (Figure 3B). As seen in Figure 15, the gradient descent estimates are typically close to the actual values used to the generate the pair of input images, although the search time is longer than in the subpixel transformation case due to the need for exhaustive search from multiple starting points. The table in Figure 16 compares actual transformation values with the estimates provided by the gradient descent method for a data set of 2-D image pairs containing one of three types of transformations (translation, rotation, or 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.

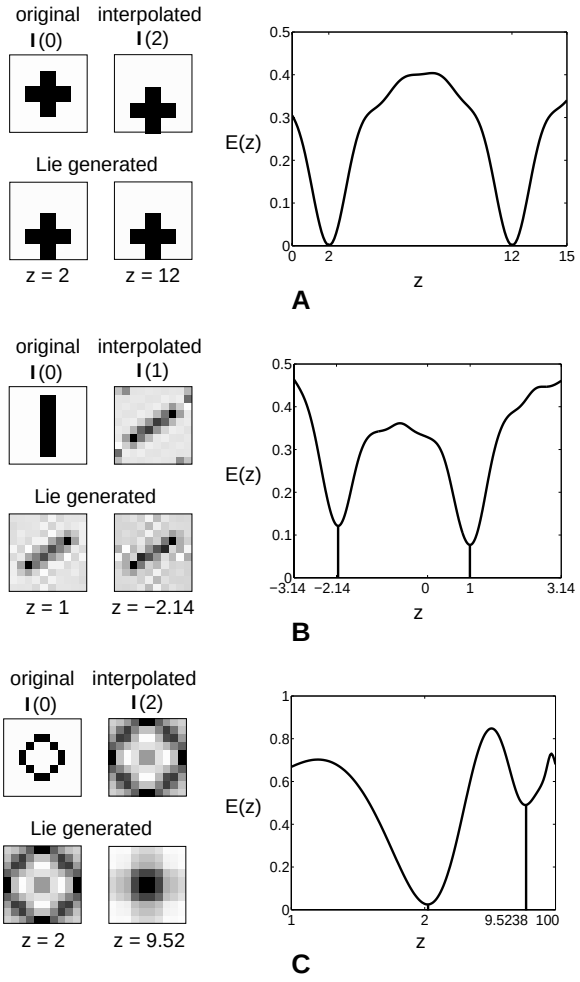

Figure 14: **Examples of Large Transformations and the Corresponding Optimization Surfaces**. (A) Example of large translation. The top two images on the left illustrate a downward translation of 2 pixels obtained by using the interpolation function in Figure 2. The plot on the right shows the error 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).

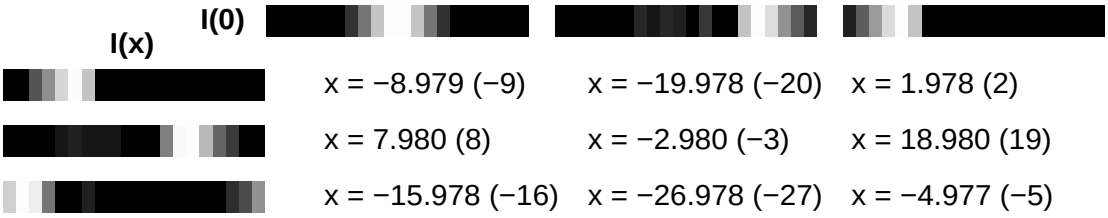

Figure 15: **Estimating Large 1-D Translations**. Comparison of estimated translation values with actual values (in parenthesis) for different pairs of reference ($\mathbf{I}(0)$) and translated images ($\mathbf{I}(x)$) shown in the form of a table.

Tranformation Type	Actual Value	Analytical Estimate (by grad. desc.)	Learned Estimate (by grad. desc.)
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

Figure 16: **Estimating Large Transformations in 2-D Images**. The table compares actual transformation values with estimates obtained using the analytically-derived and the learned Lie operator matrices for an arbitary set of image transformations.

5 Discussion and Conclusions

Our results suggest that it is possible for an unsupervised network to learn visual invariances by learning operators (or generators) for the corresponding Lie transformation groups. We demonstrated that operators for the three prominent image plane transformations, i.e., translations, rotations, and scaling, can be learned directly from training image pairs containing examples of a given transformation. The learned operators closely resemble their analytically-derived counterparts. These operators can be used in conjunction with a matrix exponential-based generative model to transform (or "warp") images by relatively large amounts. Conversely, the generative model allows estimation of transformation values directly from pairs of input images containing a given transformation.

The results presented in the paper used simple binary images to illustrate the performance of the Lie group operators. However, these operators can be applied to model transformations in arbitrary grayscale images of naturalistic scenes, as demonstrated in Figure 17. This opens up the possibility of applying the Lie group-based generative model to some of the hard problems in computer vision such as motion estimation in naturalistic scenes, simultaneous tracking and recognition of moving objects, and image stabilization. Efforts are currently underway to investigate the applicability of the model in these cases.

An important issue during estimation of large transformations based on the exponential generative model is how local minima can be avoided. 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 that is closest to zero. This is in fact already implemented

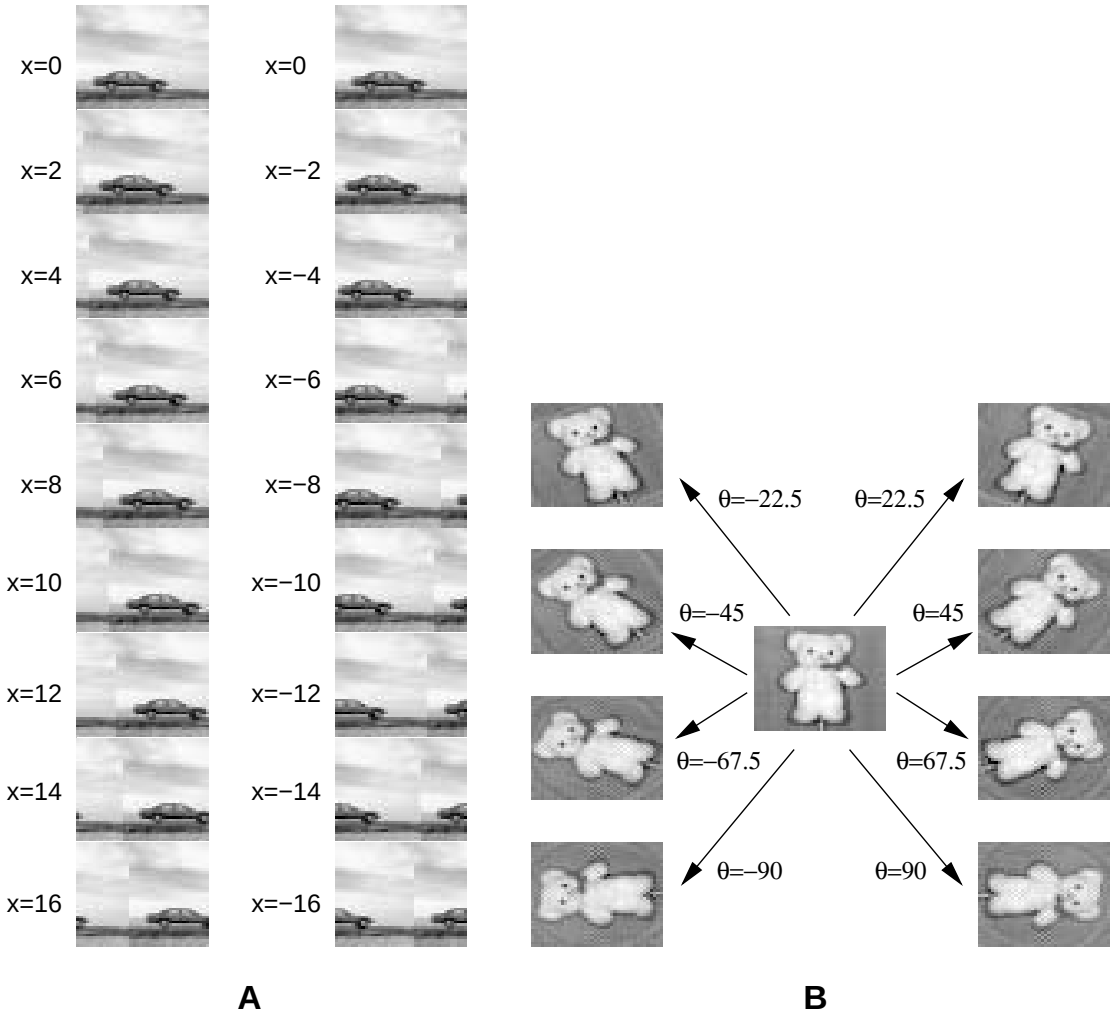

Figure 17: **Examples of Transformations on Naturalistic Grayscale Images**. (A) shows examples of using the exponential Lie generative model to translate a grayscale image of an automobile rightwards (positive values, shown on the left) and leftwards (negative values, shown on the right). (B) shows examples of rotating a grayscale image of an object clockwise (on the right) and anti-clockwise (on left) using the exponential Lie generative model.

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 at a coarse scale are used as starting points for estimating transformations at finer scales (see, for example, (Black & Jepson, 1996)). The coarse-to-fine approach may also help in alleviating the problem of noise in the learned matrices. As noted in the Results section, small amounts of noise in the learned matrices tend to get amplified for larger transformations due to the matrix exponentiation operation. A coarse-to-fine multiscale strategy could help in correcting these noise artifacts using information from higher scales.

A final research direction worthy of further study is developing more sophisticated generative models such as those that can handle multiple transformations in a given image. One potential candidate is the generative model given by: $\mathbf{I}(\mathbf{z}) = e^{\sum_{i=1}^{m} z_i G_i}\mathbf{I}(\mathbf{0}) + \mathbf{n}$, where G_i is the generator for the ith type of transformation and z_i is the value of that transformation in the input image. This model extends the Taylor-series based approach for multiple types of transformations introduced in (Rao & Ballard, 1998). Such a model could potentially allow a set of domain-specific Lie operators to be learned directly from natural video images. We expect the methods and results presented in this paper to be extremely useful in guiding our future research in this direction.

Acknowledgments

We are indebted to Dan Ruderman for providing the initial impetus for this work and for his early collaboration. This research is supported by NSF Career grant no. 133592, ONR YIP grant no. N00014-03-1-0457, and fellowships to RPNR from the Packard and Sloan foundations.

References

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.

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

Felleman, D. J. & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, *1*, 1–47.

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

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*. Houghton-Mifflin, Boston.

Grimes, D. & Rao, R. P. N. (2003). A bilinear model for sparse coding. In *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press.

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

LeCun, Y., Boser, B., Denker, J. S., Henderson, B., 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, 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. (1997). Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation*, *9*(4), 721–763.

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. & Ruderman, D. L. (1999). Learning Lie groups for invariant visual perception. In *Advances in Neural Information Processing Systems 11*, pp. 810–816. Cambridge, MA: MIT Press.

Simard, P., LeCun, Y., & Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems V*, pp. 50–58 San Mateo, CA. Morgan Kaufmann Publishers.

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.

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