Bilinear Sparse Coding for Invariant Vision

David B. Grimes and Rajesh P. N. Rao

{grimes,rao}@cs.washington.edu

Department of Computer Science and Engineering

University of Washington

Seattle, WA 98195-2350, U.S.A.

Abstract. Recent algorithms for sparse coding and independent component analysis (ICA) have demonstrated how localized features can be learned from natural images. However, these approaches do not take image transformations into account. As a result, they produce image codes that are redundant because the same feature is learned at multiple locations. We describe an algorithm for sparse coding based on a bilinear generative model of images. By explicitly modeling the interaction between image features and their transformations, the bilinear approach helps reduce redundancy in the image code and provides a basis for transformation-invariant vision. We present results demonstrating bilinear sparse coding of natural images. We also explore an extension of the model that can capture spatial relationships between the independent features of an object, thereby providing a new framework for parts-based object recognition.

1. Introduction

Algorithms for redundancy reduction and efficient coding have been the subject of considerable attention in recent years [6, 3, 4, 7, 9, 5, 12]. Although the basic ideas can be traced to the early work of Attneave [1] and Barlow [2], recent techniques such as independent component analysis (ICA) and sparse coding have helped formalize

these ideas and have demonstrated the feasibility of efficient coding through redundancy reduction. These techniques produce an efficient code by attempting to minimize the dependencies between elements of the code by using appropriate constraints.

One of the most successful applications of ICA and sparse coding has been in the area of image coding. Olshausen and Field showed that sparse coding of natural images produces localized, oriented basis filters that resemble the receptive fields of simple cells in primary visual cortex [6, 7]. Bell and Sejnowski obtained similar results using their algorithm for ICA [3]. However, these approaches do not take image transformations into account. As a result, the same oriented feature is often learned at different locations, yielding a redundant code. Moreover, the presence of the same feature at multiple locations prevents more complex features from being learned and leads to a combinatorial explosion when one attempts to scale the approach to large image patches or hierarchical networks.

In this paper, we propose an approach to sparse coding that explicitly models the interaction between image features and their transformations. A bilinear generative model is used to learn both the independent features in an image as well as their transformations. Our approach extends Tenenbaum and Freeman's work on bilinear models for learning content and style [13] by casting the problem within probabilistic sparse coding framework. Thus, whereas prior work on bilinear models used global decomposition methods such as SVD, the approach presented here emphasizes the extraction of local features by removing higher-order redundancies through sparseness constraints. We show that for natural images, this approach produces localized, oriented filters that can be translated by different amounts to account for image features at arbitrary locations. Our results demonstrate how an image can be factored into a set of basic local features and their transformations, providing a basis for transformation-

invariant vision. We conclude by discussing how the approach can be extended to allow parts-based object recognition, wherein an object is modeled as a collection of local features (or "parts") and their relative transformations.

2. Bilinear Generative Models

We begin by considering the standard linear generative model used in algorithms for ICA and sparse coding [3, 7, 9]:

$$\mathbf{z} = \sum_{i=1}^{m} \mathbf{w}_i x_i \tag{1}$$

where $\mathbf{z}$ is a k-dimensional input vector (e.g. an image), $\mathbf{w}_i$ is a k-dimensional basis vector and x_i is its scalar coefficient. Given the linear generative model above, the goal of ICA is to learn the basis vectors $\mathbf{w}_i$ such that the x_i are as independent as possible, while the goal in sparse coding is to make the distribution of x_i highly kurtotic given Equation 1.

The linear generative model in Equation 1 can be extended to the bilinear case by using two independent sets of coefficients x_i and y_i (or equivalently, two vectors $\mathbf{x}$ and $\mathbf{y}$) [13]:

$$\mathbf{z} = f(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{w}_{ij} x_i y_j \tag{2}$$

The coefficients x_i and y_j jointly modulate a set of basis vectors $\mathbf{w}_{ij}$ to produce an input vector $\mathbf{z}$. For the present study, the coefficient x_i can be regarded as encoding the presence of object feature i in the image while the y_j values determine the transformation present in the image. In the terminology of Tenenbaum and Freeman [13], $\mathbf{x}$ describes the "content" of the image while $\mathbf{y}$ encodes its "style."

Equation 2 can also be expressed as a linear equation in $\mathbf{x}$ for a fixed $\mathbf{y}$:

$$\mathbf{z} = f(\mathbf{x})_{|\mathbf{y}} = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \mathbf{w}_{ij} y_j \right) x_i = \sum_{i=1}^{m} \mathbf{w}_i^{\mathbf{y}} x_i \tag{3}$$

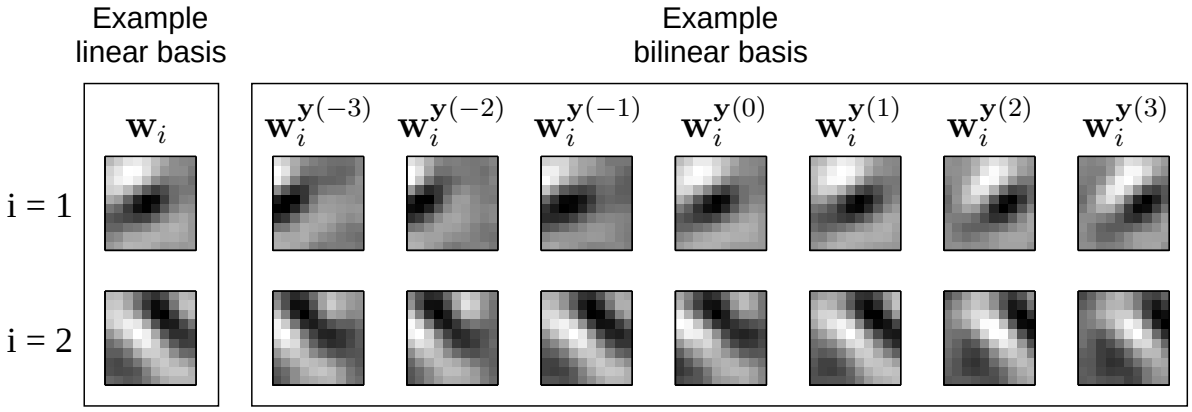

Figure 1. Examples of linear and bilinear features. A comparison of learned features between a standard linear model and a bilinear model, both trained using sparseness constraints to obtain localized, independent features. The two rows in the bilinear case depict the translated object features $\mathbf{w}_i^y$ (see Equation 3) for different $\mathbf{y}$ vectors of corresponding to translations of $-3, \ldots, 3$ pixels.

The notation $\mathbf{w}_i^{\mathbf{y}}$ signifies a transformed feature computed by the weighted sum shown above of the bilinear features $\mathbf{w}_{i,\star}$ by the values in a given $\mathbf{y}$ vector. Likewise, for a fixed $\mathbf{x}$, one obtains a linear equation in $\mathbf{y}$. Indeed this is the definition of bilinear: given one fixed factor, the model is linear with respect to the other factor. The power of bilinear models stems from the rich non-linear interactions that can be represented by varying both $\mathbf{x}$ and $\mathbf{y}$ simultaneously.

Note that the linear generative model can be seen as a special case of the bilinear model when $n = 1$, and $\mathbf{y} = 1$. A comparison between the learned features for the linear generative model (Equation 1) and the bilinear model is provided in Figure 4. Figure 1 illustrates the subsumption of the linear model, as the linear features are a single example within the range of features that can be generated by the bilinear model trained on identical data.

3. Learning Sparse Bilinear Models

3.1. Learning Bilinear Models

Our goal is to learn from image data an appropriate set of basis vectors $\mathbf{w}_{ij}$ that effectively describe the interactions between the feature vector $\mathbf{x}$ and the transformation vector $\mathbf{y}$. A commonly used approach in unsupervised learning is to minimize the sum of squared pixel-wise errors over all images:

$$E_1(\mathbf{w}_{ij}, \mathbf{x}, \mathbf{y}) = ||\mathbf{z} - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{w}_{ij} x_i y_j||^2 \tag{4}$$

$$= (\mathbf{z} - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{w}_{ij} x_i y_j)^T (\mathbf{z} - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{w}_{ij} x_i y_j) \tag{5}$$

where $||\cdot||$ denotes the L_2 norm of a vector. A standard approach to minimizing such a function is to use gradient descent and alternate between minimization with respect to $\{\mathbf{x}, \mathbf{y}\}$ and minimization with respect to $\mathbf{w}_{ij}$. Unfortunately, the optimization problem as stated is underconstrained. The function E_1 has many local minima and results from our simulations indicate that convergence is difficult in many cases. There are many different ways to represent an image, making it difficult for the method to converge to a basis set that can generalize effectively.

A related approach is presented by Tenenbaum and Freeman [13]. Rather than using gradient descent, their method estimates the parameters directly by computing the singular value decomposition (SVD) of a matrix A containing input data corresponding to each content class $\mathbf{x}$ in every style $\mathbf{y}$. Their approach can be regarded as an extension of methods based on principal component analysis (PCA) applied to the bilinear case. The SVD approach avoids the difficulties of convergence that plague the gradient descent method and is much faster in practice. Unfortunately, the learned features tend to be global and non-localized similar to those obtained from PCA-based methods based on second-order statistics. As a result, the method is unsuitable for the problem of learning

local features of objects and their transformations.

The underconstrained nature of the problem can be remedied by imposing constraints on $\mathbf{x}$ and $\mathbf{y}$. In particular, we could cast the problem within a probabilistic framework and impose specific prior distributions on $\mathbf{x}$ and $\mathbf{y}$ with higher probabilities for values that achieve certain desirable properties. We focus here on the class of sparse prior distributions for several reasons: (a) by forcing most of the coefficients to be zero for any given input, sparse priors minimize redundancy and encourage statistical independence between the various x_i and between the various y_j [7], (b) there is growing evidence for sparse representations in the brain – the distribution of neural responses in visual cortical areas is highly kurtotic i.e. the cell exhibits little activity for most inputs but responds vigorously for a few inputs, causing a distribution with a high peak near zero and long tails, (c) previous approaches based on sparseness constraints have obtained encouraging results [7], and (d) enforcing sparseness on the x_i encourages the parts and local features shared across objects to be learned while imposing sparseness on the y_j allows object transformations to be explained in terms of a small set of basic transformations.

3.2. Probabilistic Bilinear Sparse Coding

Our probabilistic model for bilinear sparse coding follows a standard Bayesian MAP (maximum a posteriori) approach. Thus we begin by factoring the conditional likelihood $P(x_i, y_i|z_i)$ as such:

$$P(\mathbf{x}, \mathbf{y}, \mathbf{w}_{ij}|\mathbf{z}) = \frac{P(z_i|x_i, y_i, \mathbf{w}_{ij})P(x_i)P(y_i)}{P(z_i)} \tag{6}$$

$$\propto P(z_i|x_i, y_i, \mathbf{w}_{ij})P(x_i)^{\alpha}P(y_i)^{\beta} \tag{7}$$

Eq. 7 makes two modifications to Eq. 6 Firstly, the observation (image) likelihood prior is assumed uniform and thus ignored. Secondly, in order to study the effects of the priors

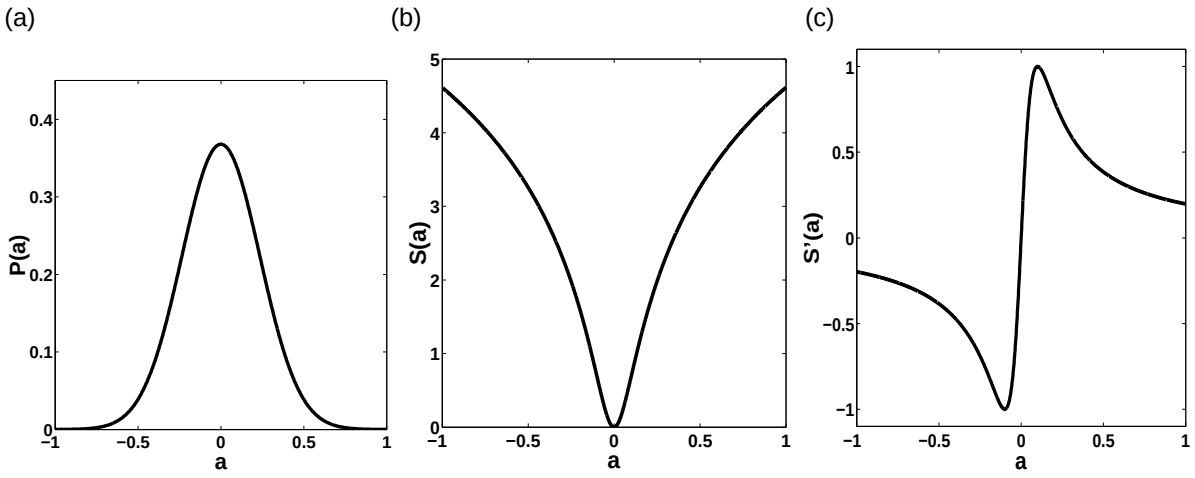

Figure 2. A probabilistic sparse coding prior. (a) shows the probability distribution function for the Cauchy sparse coding prior. Although the distribution appears similar to a Gaussian distribution, the Cauchy is super-Gaussian (highly kurtotic). (b) shows the derived sparseness error function. (c) illustrates the non-linearity introduced in the derivative of the sparseness function. Note that the function differentially forces small coefficients towards zero, and only at some threshold are large coefficients made larger.

on the bilinear basis, we allow the priors for both $\mathbf{x}$ and $\mathbf{y}$ to be weighted arbitrarily by the respective factors α and β.

We assume the following priors for x_i and y_j:

$$P(x_i) = \frac{1}{Q_\alpha} e^{-\alpha S(x_i)} \tag{8}$$

$$P(y_j) = \frac{1}{Q_\beta} e^{-\beta S(y_j)} \tag{9}$$

where Q_α and Q_β are normalization constants, α and β are parameters that control the degree of sparseness, and S is a "sparseness function." For this study, we used $S(a) = \log(1 + a^2)$. As shown in Figure 2, our choice of $S(a)$ corresponds to a Cauchy prior distribution, which exhibits a useful non-linearity in the derivative $S'(a)$.

Within a probabilistic framework, the squared error function E_1 summed over all images can be interpreted as representing the negative log likelihood of the data given the parameters: $-\log P(\mathbf{z}|\mathbf{w}_{ij}, \mathbf{x}, \mathbf{y})$ (see, for example, [7]). The priors $P(x_i)$ and $P(y_j)$ can be used to marginalize this likelihood to obtain the new likelihood function:

$L(\mathbf{w}_{ij}) = P(\mathbf{z}|\mathbf{w}_{ij})$. The goal then is to find the $\mathbf{w}_{ij}$ that maximize L, or equivalently, minimize the negative log of L. Under certain reasonable assumptions (discussed in [7]), this is equivalent to minimizing the following optimization function over all input images:

$$E(\mathbf{w}_{ij}, \mathbf{x}, \mathbf{y}) = ||\mathbf{z} - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{w}_{ij} x_i y_j||^2 + \alpha \sum_{i=1}^{m} S(x_i) + \beta \sum_{j=1}^{n} S(y_j) \tag{10}$$

The gradient of E can be used to derive update rules for the components x_a and y_b of the feature vector $\mathbf{x}$ and transformation vector $\mathbf{y}$ respectively for any image $\mathbf{z}$, assuming a fixed basis $\mathbf{w}_{ij}$:

$$\frac{dx_a}{dt} = -\frac{1}{2}\frac{\partial E}{\partial x_a} = \sum_{q=1}^{n} \mathbf{w}_{aq}^T (\mathbf{z} - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{w}_{ij} x_i y_j) y_q + \frac{\alpha}{2} S'(x_a) \tag{11}$$

$$\frac{dy_b}{dt} = -\frac{1}{2}\frac{\partial E}{\partial y_b} = \sum_{q=1}^{m} \mathbf{w}_{qb}^T (\mathbf{z} - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{w}_{ij} x_i y_j) x_q + \frac{\beta}{2} S'(y_b) \tag{12}$$

Given a training set of inputs $\mathbf{z}_l$, the values for $\mathbf{x}$ and $\mathbf{y}$ for each image after convergence can be used to update the basis set $\mathbf{w}_{ij}$ in batch mode according to:

$$\frac{d\mathbf{w}_{ab}}{dt} = -\frac{1}{2}\frac{\partial E}{\partial \mathbf{w}_{ab}} = \sum_{l=1}^{q} (\mathbf{z}_l - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{w}_{ij} x_i y_j) x_a y_b \tag{13}$$

3.3. Algorithm for Learning Model of Translating Image Patches

We now present our algorithm which utilizes the above gradients with the goal of learning localized features in natural image patches and their transformations with respect to two-dimensional translations. A large class of algorithms is possible if one were to minimize E by changing more than one variable per gradient step. Our experience shows that obtaining convergence reliably is rather difficult in this situation. Fortunately it appears unnecessary to minimize with respect to more than one variable at a time, and further simply minimizing E with respect to a single variable until near convergence yields good results, particularly when combined with a batch derivative approach. In

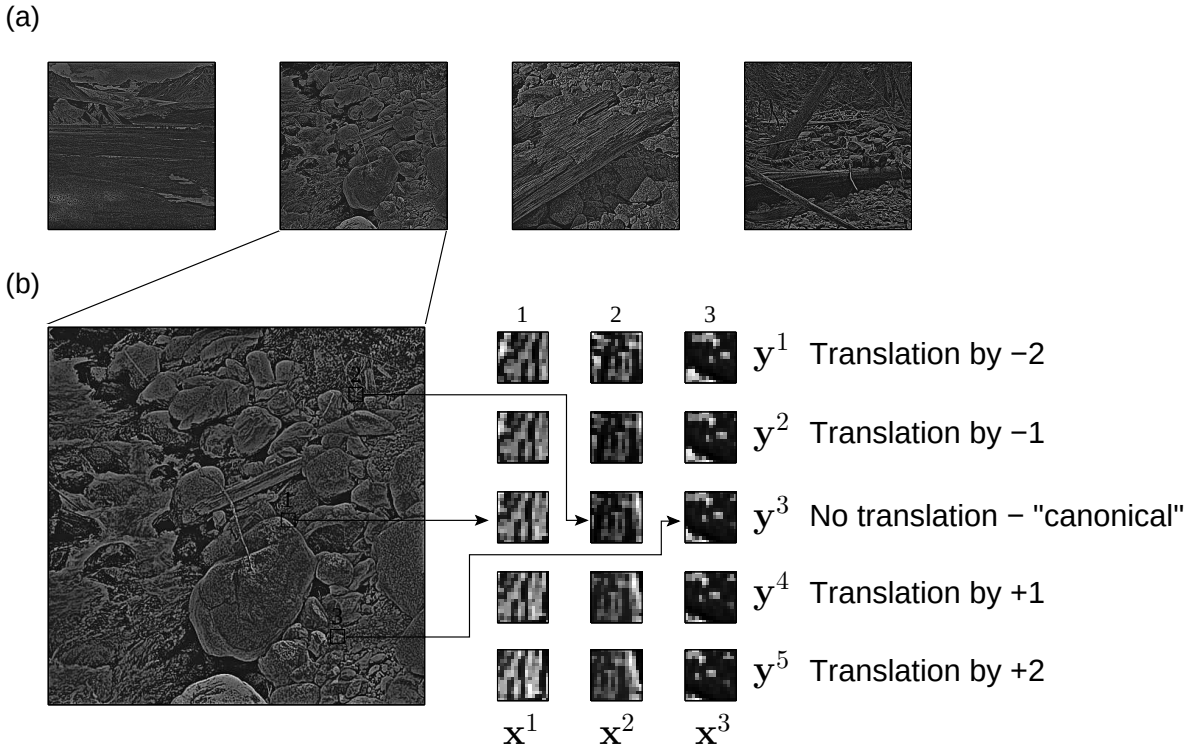

Figure 3. Example training data from natural scene images. Training data is formed by randomly selected patch locations from a set of natural images (a). As shown in (b) the patch location is then transformed, in this example we use horizontal translations of ± 2. To achieve style/content separation a single $\mathbf{x}$ vector is assigned to represent each column, and a $\mathbf{y}$ vector to each row.

our implementation we use a conjugate gradient method to speed the convergence in minimizing E with respect to $\mathbf{x}$ and $\mathbf{y}$.

Our algorithm enforces style and content separation by iteratively holding $\mathbf{x}$, then $\mathbf{y}$ fixed in a particular order related to the presentation of the images (see Figure 3). This separation seeks to minimize the effects of style transformations on $\mathbf{x}$, and similarly the effect of content changes on $\mathbf{y}$. We first initialize a matrix Y which contains a set of vectors describing each style. In order to regularize Y and avoid overfitting any particular patch (or set of patches) we slowly adapt Y over time. Specifically we find an initial $\mathbf{x}$ vector that describes the content of a randomly selected patch, then for each transformation adapt the corresponding $\mathbf{y}$ vector slowly based on the parameter ϵ.

Finding the initial $\mathbf{x}$ vector relies on having a reference $\mathbf{y}$. Thus we refer to one of the transformations as "canonical", often the identity transformation. This special $\mathbf{y}_{can}$ is thus used for bootstrapping, but besides this use is adapted exactly like the rest of the style vectors in Y.

One difficulty in the sparse coding formulation of Eq. 10 is that the algorithm can trivially minimize the sparseness function by making $\mathbf{x}$ or $\mathbf{y}$ very small and increasing the $\mathbf{w}_{ij}$ basis vector norms to maintain the desired output range. Therefore as suggested by Olshausen and Field [7], in order to keep the basis vectors from growing without bound, we adapt the L_2 norm of each basis vector in such a way that the variances of the x_i and y_j were maintained at a fixed desired level (σ_g^2). Simply forcing the basis vectors to have a certain norm can lead to instabilities, thus a more robust method "soft" variance normalization is employed. The element-wise variance of the $\mathbf{x}$ vectors is tracked in the vector $\mathbf{x}_{var}$ and adapted according to the parameter ε. A gain term (see algorithm lines 23-24) $\mathbf{x}_{gain}$ indicates a multiplicative factor for adapting the norm of a particular basis vector. An additional complication in the bilinear case is that $||\mathbf{w}_{ij}||_2$ is related to the variance of both x_i and y_j. One method we used was to compute the joint gain J as the geometric mean of the elements in the gain vectors $\mathbf{x}_{gain}$ and $\mathbf{y}_{gain}$. However, if sparseness is desired for $\mathbf{x}$ but not $\mathbf{y}$ the variance of $\mathbf{y}$ will rapidly increase, as the variance of $\mathbf{x}$ rapidly increases, and no changes of the length of basis vectors $\mathbf{w}_{ij}$ will solve this problem. Thus we developed a method which performs soft variance normalization directly on the evolving $\mathbf{y}$ vectors, and bases the scaling of the basis vectors only on the variance of $\mathbf{x}$.

In the following algorithm all capital letters indicate matrices containing column vectors, indexed using Matlab style slicing. I is a matrix containing all training images arranged into columns. T is $2 \times r$ matrix containing the translation parameters for each

r transformations to be learned. The $\star$ indicates the element wise product. *cidx* denotes the index of the canonical transformation.

LearnSparseBilinearModel$(I, T, \alpha, \beta, \epsilon, \varepsilon, \eta, \gamma, \sigma_g^2)$

$W \leftarrow$ RandNormalizedVectors(k, m, n)
$Y \leftarrow$ RandNormalizedVectors(n, r)
$\mathbf{y}_{can} \leftarrow Y(:, cidx)$
for $iter \leftarrow 1$ **to** $maxIter$
    $P \leftarrow$ SelectPatchLocations(I, q)
    $Z_{can} \leftarrow$ ExtractPatches(I, P)
    $X \leftarrow$ CGSparseFit$(W, \mathbf{y}_{can}, Z_{can}, \alpha)$
    $dW \leftarrow$ Zeros(k, m, n)
    for $i \leftarrow 1$ **to** r
        $Z \leftarrow$ ExtractPatches$(I,$ Transform$(P, T(:, i)))$
        $Y_{batch} \leftarrow$ CGSparseFit(W, X, Z, β)
        $Y(:, i) \leftarrow (1 - \epsilon)Y(:, i) + \epsilon \cdot$ SampleMean(Y_{batch})
        $\mathbf{y}_{var} \leftarrow (1 - \varepsilon)\mathbf{y}_{var} + \varepsilon \cdot$ SampleVar(Y_{batch})
        $\mathbf{y}_{gain} \leftarrow \mathbf{y}_{gain} \star \left(\frac{\mathbf{y}_{var}}{\sigma_g^2}\right)^{\gamma}$
        $Y(:, i) \leftarrow$ NormalizeVectors$(Y(:, i), \mathbf{y}_{gain})$
        if $i = cidx$
            then $\mathbf{y}_{can} \leftarrow Y(:, cidx)$
        $dW \leftarrow dW + \left(\frac{1}{r}\right) dEdW(Z, W, X, Y(:, i))$
    $W = W + \eta dW$
    $\mathbf{x}_{var} \leftarrow (1 - \varepsilon)\mathbf{x}_{var} + \varepsilon \cdot$ SampleVar(X)
    $\mathbf{x}_{gain} \leftarrow \mathbf{x}_{gain} \star \left(\frac{\mathbf{x}_{var}}{\sigma_g^2}\right)^{\gamma}$
    $W \leftarrow$ NormalizeVectors$(W, \mathbf{x}_{gain})$

4. Results

4.1. Training Methodology

We tested the algorithms for bilinear sparse coding on natural image data. The natural images we used are distributed by Olshausen and Field [7], along with the code for their algorithm. The training set of images consisted of 10×10 patches randomly extracted from ten 512×512 source images. The images are pre-whitened to equalize large variances in frequency, and thus speed convergence. We choose to use a complete basis where $m = 100$ and we let n be somewhat less than the number of transformations (including the no-transformation case). The sparseness parameters α and β were set to approximately 20 and 1.5, and the effects of these parameters are discussed in further detail later. In order to assist convergence all learning occurs in batch mode, where the batch consisted of $q = 100$ image patches. The W step size η for gradient descent using Equation 13 was set to 0.05. Variance normalization used $\varepsilon = 0.25$, and $\gamma = 0.05$, and $\sigma_g^2 = 0.1$. These parameters are not necessarily the optimum parameters, and the algorithm is robust with respect to significant parameters changes. Generally only the amount of time required to find a good sparse representation changes with untuned parameters. In the case presented here the algorithm converged after approximately 2500 iterations. The transformations were chosen to be 2D translations in the range $[-4 : 4]$ pixels in both the axes. The style/content separation was enforced by learning a single $\mathbf{x}$ vector to describe an image patch regardless of its translation, and likewise a single $\mathbf{y}$ vector to describe a particular style given any image patch content.

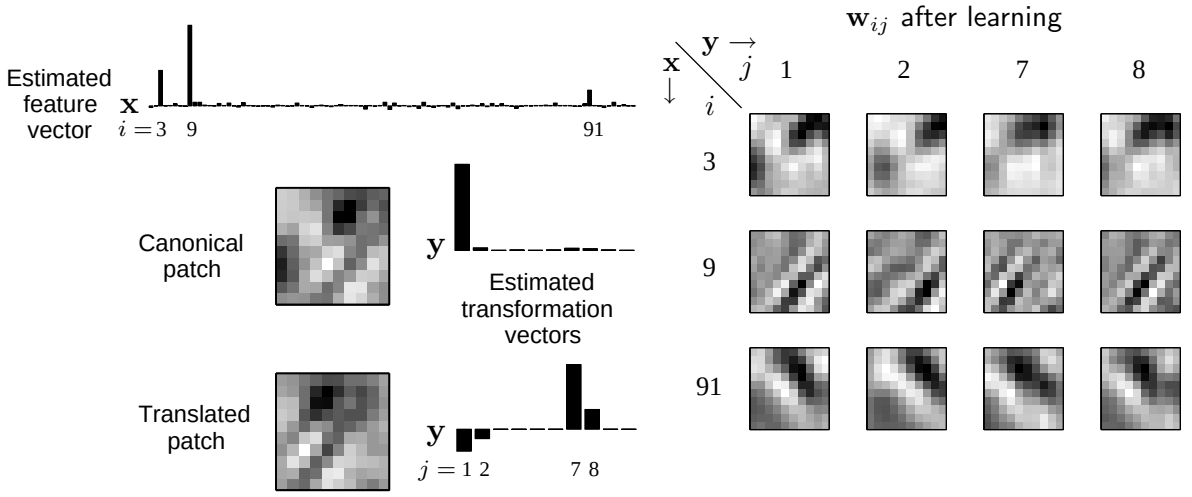

Figure 4. Representing natural images and their transformations with a sparse bilinear model. The representation of an example natural image patch, and of the same patch translated to the left. Note that the bar plot representing the $\mathbf{x}$ vector is indeed sparse, having only three significant coefficients. The code for the style vectors for both the canonical patch, and the translated one is likewise sparse. The $\mathbf{w}_{ij}$ basis images are shown for those dimensions which have non-zero coefficients for x_i or y_j.

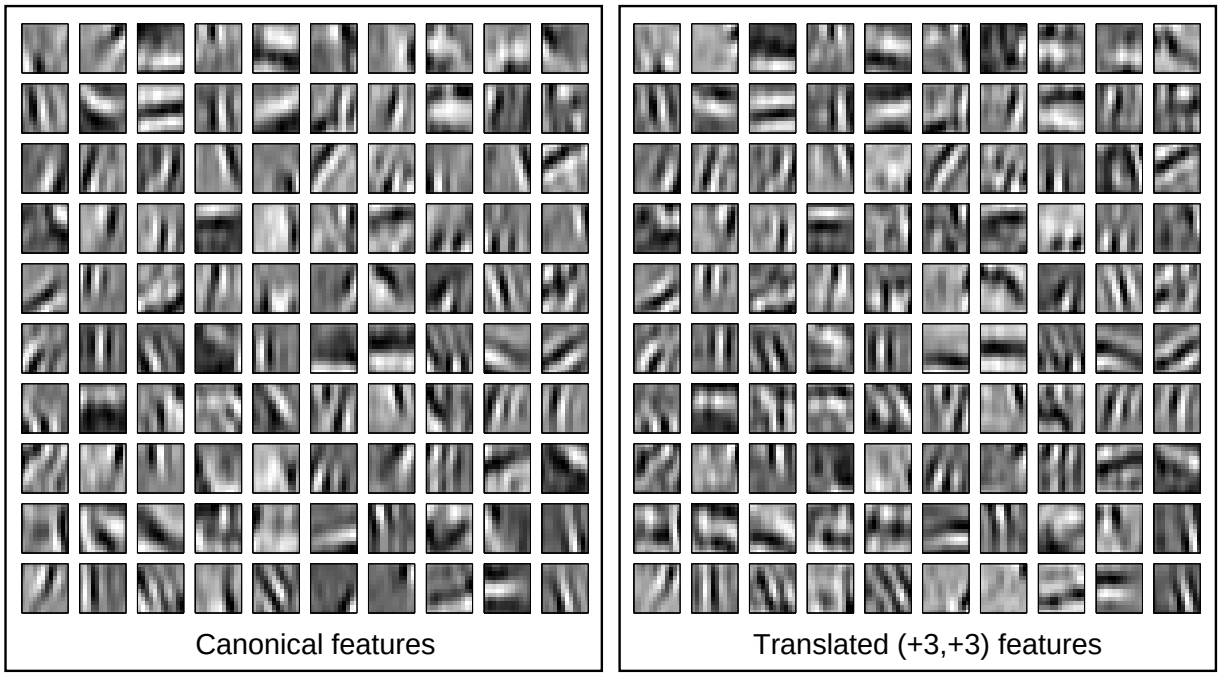

Figure 5. Localized, oriented set of learned basis features. The full 100 features learned with the sparse bilinear model. A sample translation of three pixels in both dimensions is also shown.

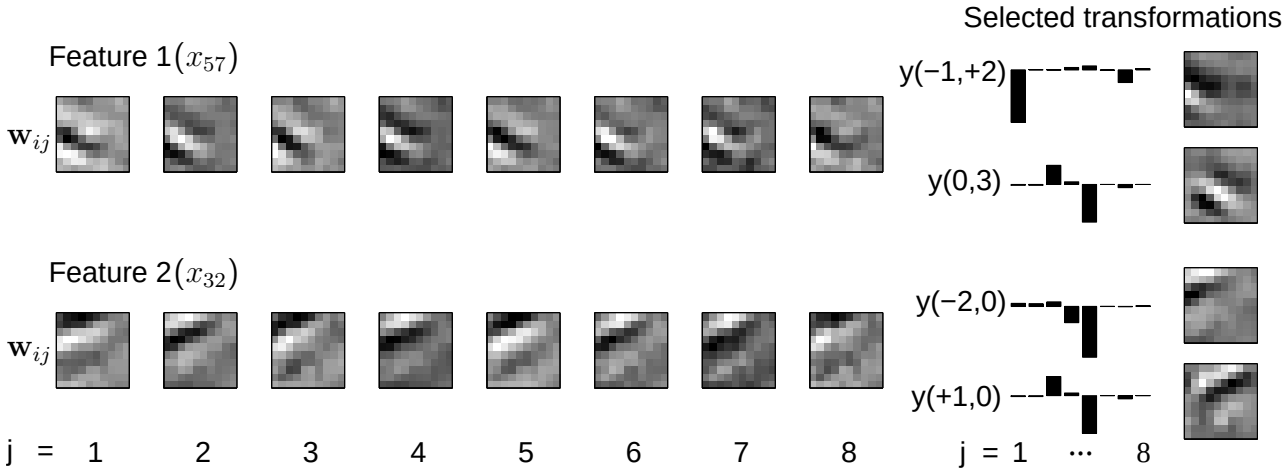

Figure 6. Translating a learned feature to multiple locations. The two rows of eight images represent the individual basis vectors $\mathbf{w}_{ij}$ for two values of i. The y_j values for two selected transformations for each i are shown as bar plots. $y(a, b)$ denotes a translation of (a, b) pixels in the Cartesian plane. The last column shows the resulting basis vectors after translation.

4.2. Bilinear Sparse Coding of Natural Images

Figures 4 and 5 show the results of training on natural image data. Both show simple, localized, and oriented features, and the bilinear method is able to model the same features under different transformations. In this case, the range $[-3, 3]$ horizontal translations were used in the training of the bilinear model. Figure 4 provides an example of how the bilinear sparse coding model encodes a natural image patch and the same patch after it has been translated. Note that in this case both the $\mathbf{x}$ and $\mathbf{y}$ vectors are sparse.

Figure 6 shows how the model can account for a given localized feature at different locations by varying the $\mathbf{y}$ vector. As shown in the last column of the figure, the translated local feature is generated by linearly combining a sparse set of basis vectors $\mathbf{w}_{ij}$.

4.3. Effects of Sparseness on Representation

The free parameters α and β play an important role in deciding how sparse the coefficients in the vectors $\mathbf{x}$ and $\mathbf{y}$ are. Likewise, the sparsity of the vectors is intertwined with the desired local and independent properties of the $\mathbf{w}_{ij}$ bilinear basis features. As noted in other research on sparsity [6] both the attainable sparseness and independence of features also depend on the model dimensionality, in our case the parameters m and n. In all of our experiments we use a complete basis (in which $m = k$) for content representation, assuming that the style transformations do not affect the number of basis needed for representation. We believe this is justified also by the very idea that changes in style should not change the intrinsic content.

However, might also desire that the $\mathbf{y}$ style vector use a sparse representation as well. In the case of affine transformations using a complete basis for $\mathbf{y}$ means using a large value on n. From a practical perspective this is undesirable as it would essentially equate to the tiling of features at all possible transformations. Thus in most of our experiments we use low values or zero for the value of β, and set n low (between 4 and 8). This setup allows for learning sparse, independent content features while taking advantage of dimensionality reduction in the coding of transformations.

Setting α requires some additional analysis. Ideally we desire to maximize statistical independence of the bilinear basis. Figure 7 illustrates the effect of the spareness likelihood weighting term β on the sparseness error for the $\mathbf{x}$ content representation. For each value of β ten trials of the learning algorithm. The sparseness error shown is the sum of the sparseness function $S(x_i j)$ over all m dimensions of 1000 vectors generated from a random sample of natural image patches, after training. Figure 8 illustrates how the sparsity, kurtosis, and obtaining localized, independent basis vectors are related.

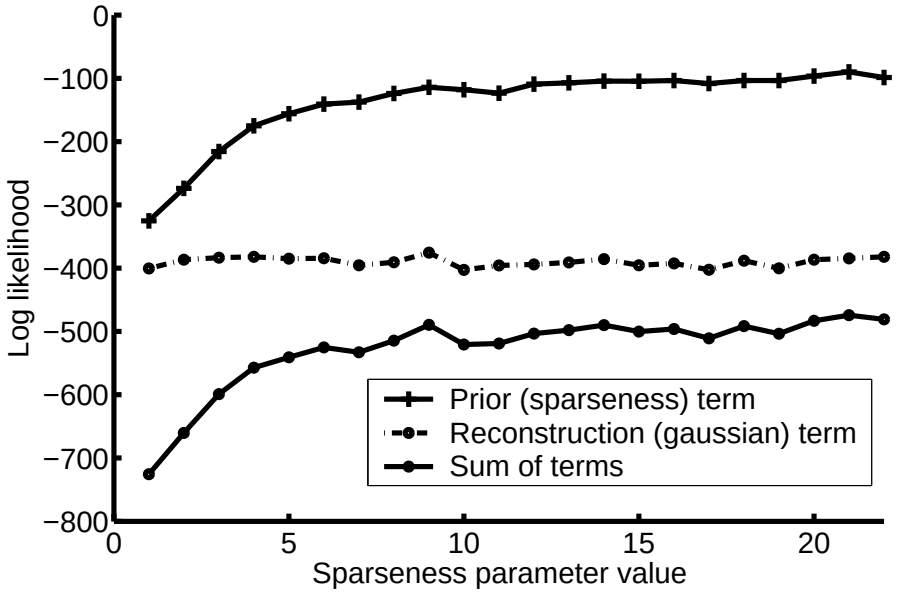

Figure 7. Sparseness weighting term vs. sparseness. As the sparseness term in the likelihood function is more highly weighted than the reconstruction term the sparseness error is reduced. The reduction in error represents a more efficient core characterized by less redundancy in features. Note that the reconstruction error is effectively unchanged, even as the spareness term is weighted twenty times more heavily, thus illustrating that the sparse code doesn't result in a loss of representational fidelty.

4.4. Style and Content Invariance

A series of experiments were conducted to analyze the invariance properties of the sparse bilinear model. These experiments looked at how the style and content representations change when the actual style and content of the input changes. In this case we looked at how shifting and selecting various patches affects the $\mathbf{x}$ and $\mathbf{y}$ vectors. Since $\mathbf{x}$ and $\mathbf{y}$ are arbitrary dimensional vectors we used the norm of the vector difference as our metric for representation change. The input is transformed (smoothly in the case of style, discretely in the case of content) from one iteration to the next, and the change is measured as:

$$\Delta\mathbf{x_i} = |\mathbf{x}_i - \mathbf{x}_{i-1}| \qquad \Delta\mathbf{y_i} = |\mathbf{y}_i - \mathbf{y}_{i-1}| \tag{14}$$

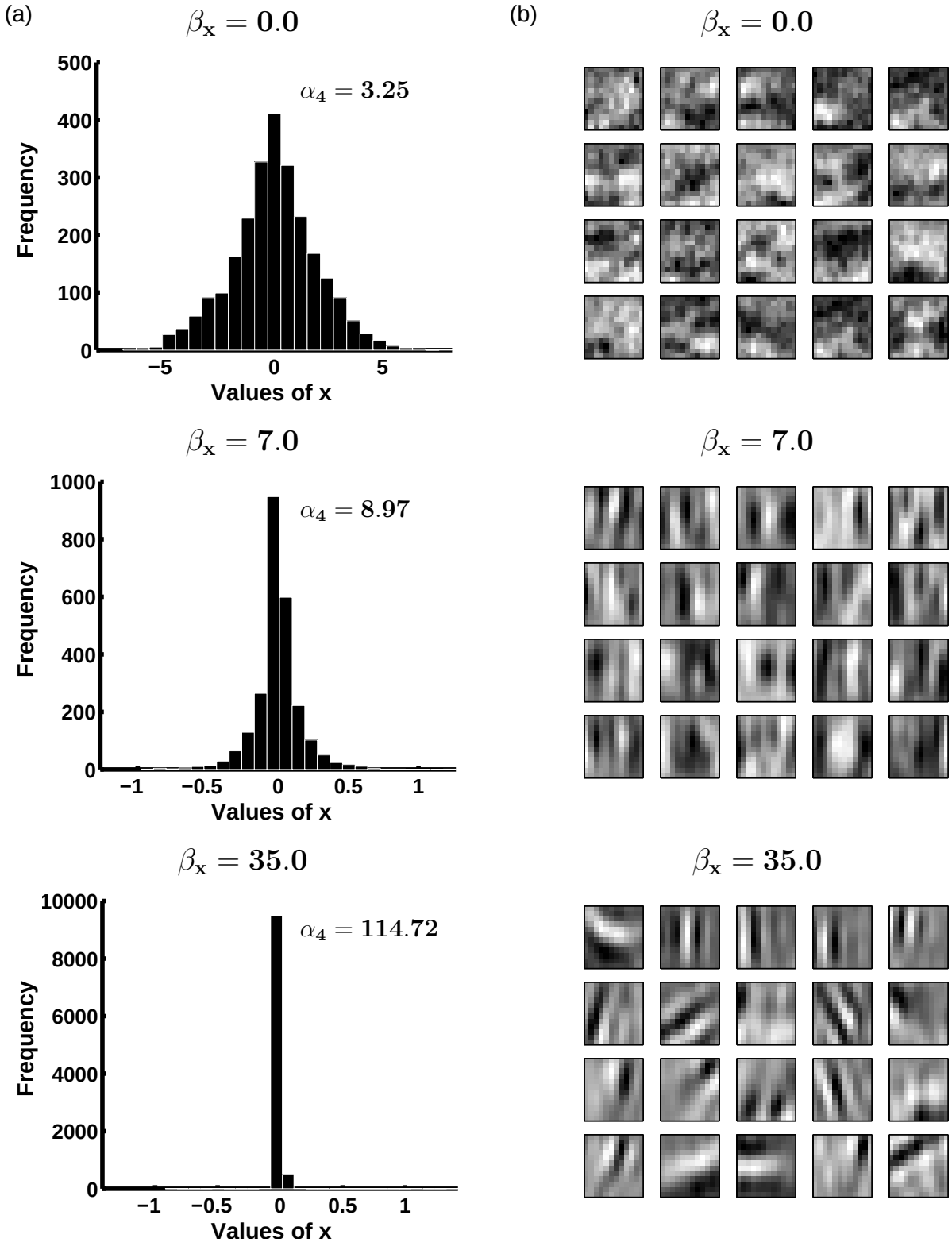

Figure 8. Sparseness prior yields highly kurtotic coefficient distributions. The effect of strongly weighting the sparseness prior on the (a) kurtosis (denoted α_4) of x_i coefficient distribution, and (b) a subset of the corresponding bilinear basis vectors.

Figure 9 shows the result of vertically shifting the patch location and recording the subsequent representation changes. Likewise figure 10 shows the result of selecting random patches and translating each in an identical way. Both figures show a strong degree of invariance of representation. These previous figures look at sample runs, and give a qualitative result of the invariance properties of the model. Figures 11 and 11 quantify (on average after 100 runs) how much changing style affects content representation, and vice versa.

4.5. Interpolation for Continuous Translations

Although transformations are learned as discrete style classes, we found that the sparse bilinear model can handle continuous transformations. Interpolation in the style space can obtain a representation of some content at a translation in between two discrete translations that were learned during initial model training. Figure 13(a) shows the values of the six dimensional $\mathbf{y}$ vector for each learned translation. The circular point are values that were learned in model training, plus markers indicate interpolated style vectors. Note that for the most part the coefficients vary smoothly with respect to the translation amount. This property allows simple linear interpolation between styles, similar to the method of locally linear embedding [11]. As sub-pixel translations are difficult to illustrate, a model was trained with only the transformations $-4, -2, 0, +2, +4$. The style representation vectors learned are shown in Figure 13(b). Figure 13(c) shows the reconstructed patches for both neighboring styles and the interpolated style, as well as mean squared error (MSE) values for the reconstructions. Note that while the interpolation MSE values are somewhat higher than the reconstructions at learned translations, they are still in the same neighborhood.

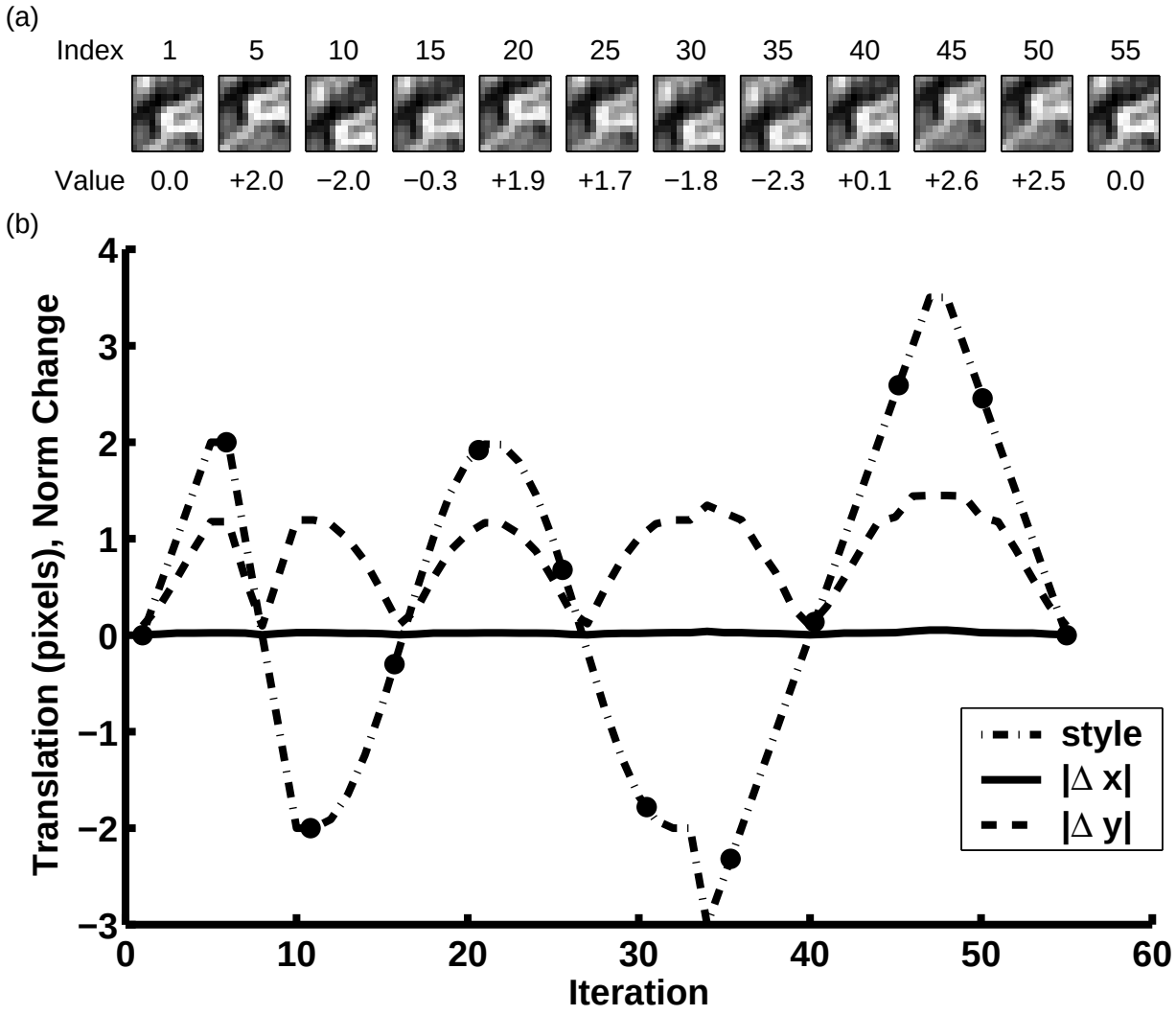

Figure 9. Transformation invariance property of the sparse bilinear model. (a) shows a randomly selected natural image patch transformed by an arbitrary nonlinear sequence of vertical translations. (b) shows the effects of the transformations on the style ($\mathbf{y}$) and content ($\mathbf{x}$) vectors. Note that the magnitude of the change in $\mathbf{y}$ is directly correlated to the magnitude of the vertical translation, while the change in $\mathbf{x}$ is insignificant (mean $|\Delta\mathbf{x}| = 0.04$), thus illustrating the transformation invariance property of the bilinear model.

4.6. Towards Parts-Based Object Recognition

The bilinear generative model in Equation 2 uses the same set of transformation values y_j for all the features $i = 1, \ldots, m$. Such a model is appropriate for global transformations that apply to an entire image region such as a shift of p pixels for an image patch or a global illumination change.

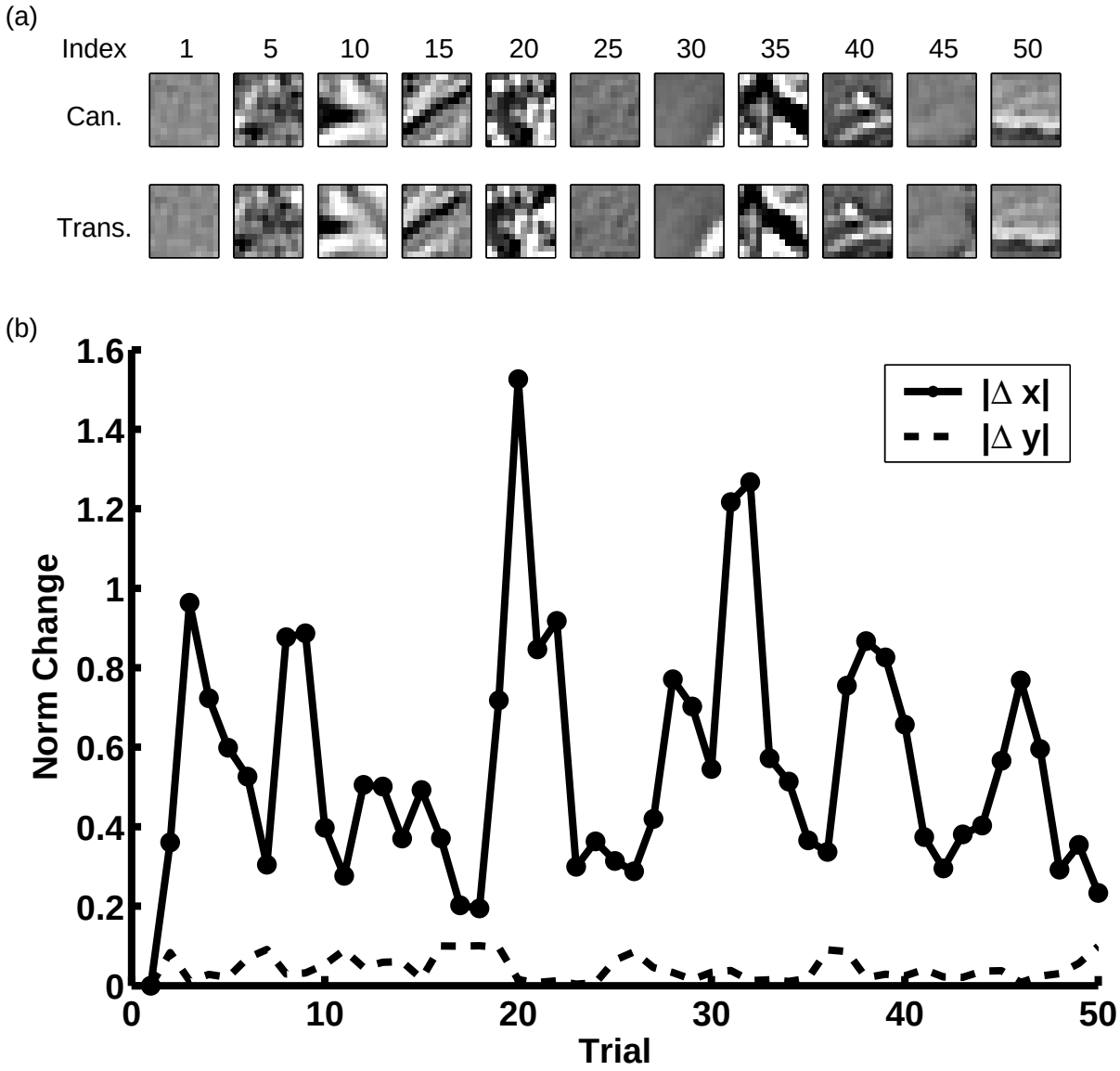

Figure 10. Invariance of style representation to content. (a) shows a sequence of randomly selected patches (denoted "Can.") and their horizontally shifted versions (denoted "Trans."). (b) shows the change in the estimated $\mathbf{y}$ for the translated version of each patch. Note that the content representation fluctuates wildly, while the style vector changes very little.

Consider the problem of representing an object in terms of its constituent parts. In this case, we would like to be able to transform each part independently of other parts in order to account for the location, orientation, and size of each part in the object

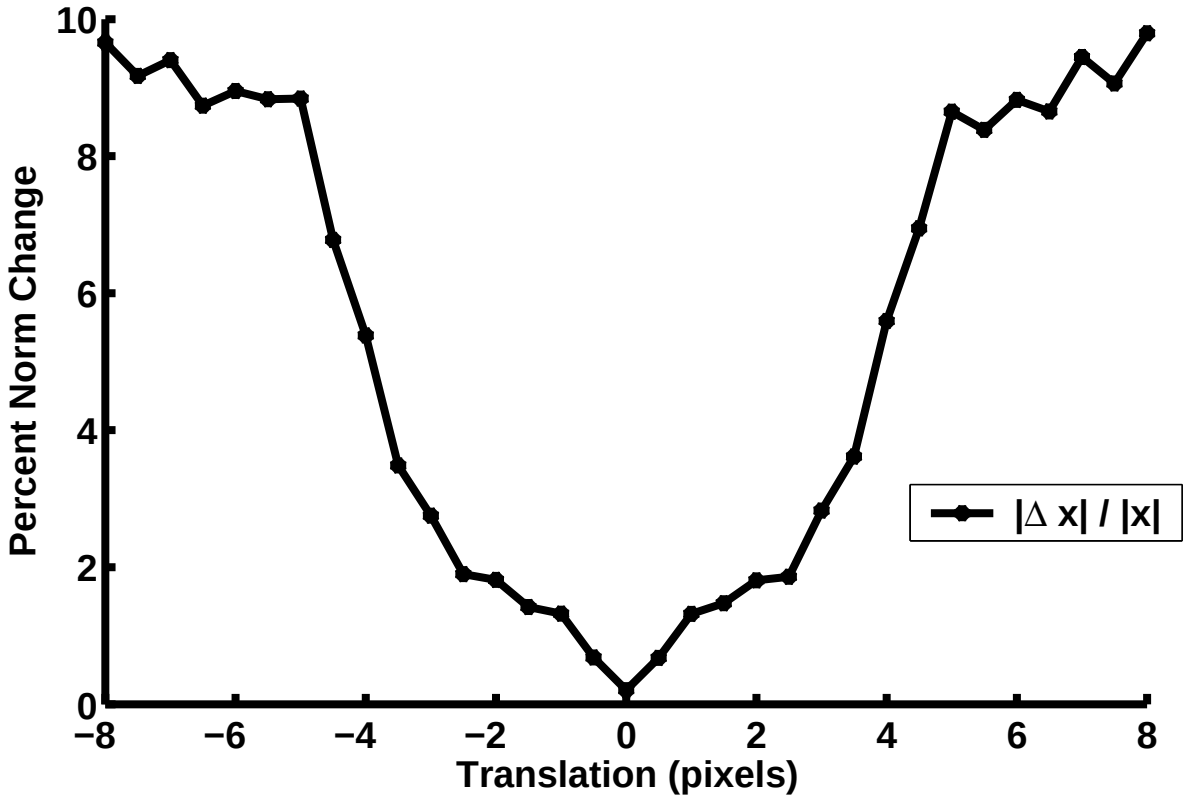

Figure 11. Translational effects on content representation. The average relative change in $\mathbf{x}$ versus translation over 100 experiments in which image 12x12 patches were shifted by varying degress. Note that translations in the range $(-3, +3)$ the relative change in $\mathbf{x}$ is small, yet as more and more features are shifted into the patch the representation must change.

image. The standard bilinear model can be extended to address this need as follows:

$$\mathbf{z} = \sum_{i=1}^{m} (\sum_{j=1}^{n} \mathbf{w}_{ij} y_j^i) x_i \tag{15}$$

Note that each object feature i now has its own set of transformation values y_j^i. The double summation is thus no longer symmetric. Also note that the standard model (Equation 2) is a special case of Equation 15 where $y_j^i = y_j$ for all i.

We have conducted preliminary experiments to test the feasibility of Equation 15 using a set of object features learned for the standard bilinear model. Figure 14 shows the results. These results suggest that allowing independent transformations for the different features provides a rich substrate for modeling images and objects in terms of a set of local features (or parts) and their individual transformations.

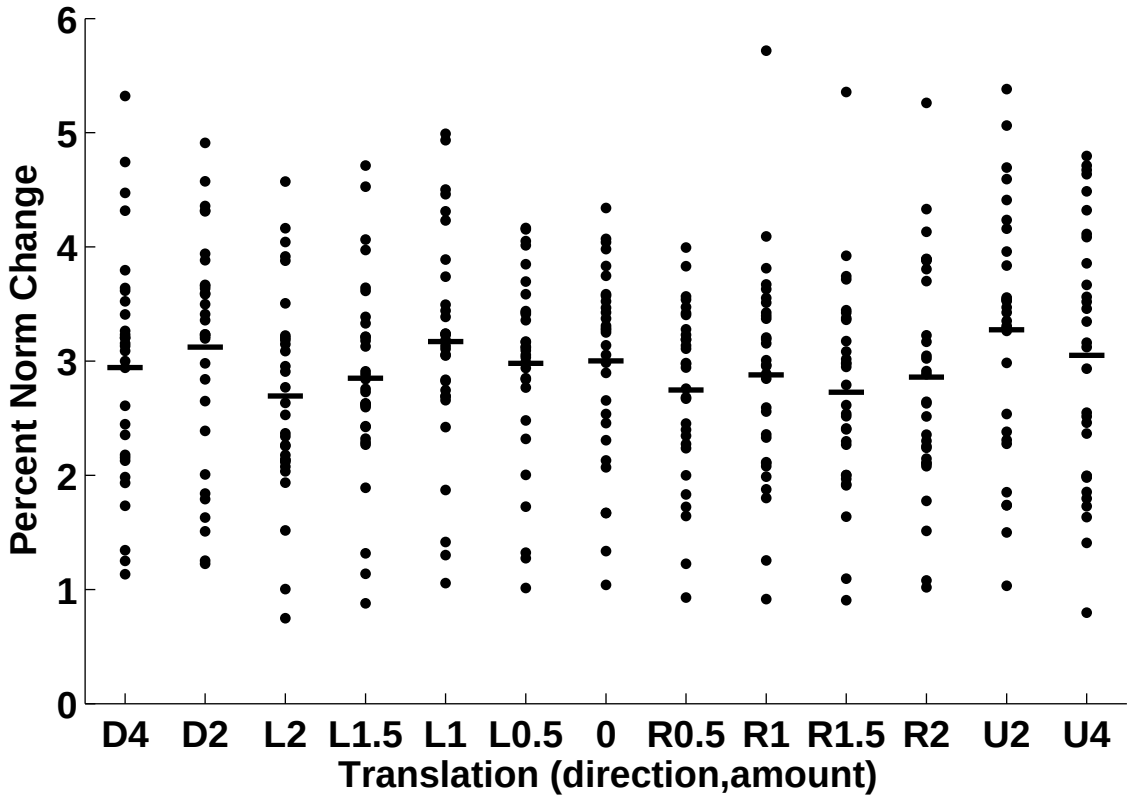

Figure 12. Changing content effects on style representation. The average relative change in $\mathbf{y}$ for random patch content over 100 experiments in which various transformations were performed on a randomly selected patch. No discernible pattern seems to exist which would suggest that some transformations are more sensitive to content. The bar shows the mean relative change for each transformation.

5. Related Work and Discussion

FIXME, Insert bunches of references here.

6. Summary and Conclusion

A fundamental problem in vision is to simultaneously recognize objects and their transformations [8, 10]. Bilinear generative models provide a tractable way of addressing this problem by factoring an image into object features and transformations using a bilinear equation. Previous approaches used unconstrained bilinear models and produced global basis vectors for image representation [13]. In contrast, recent research

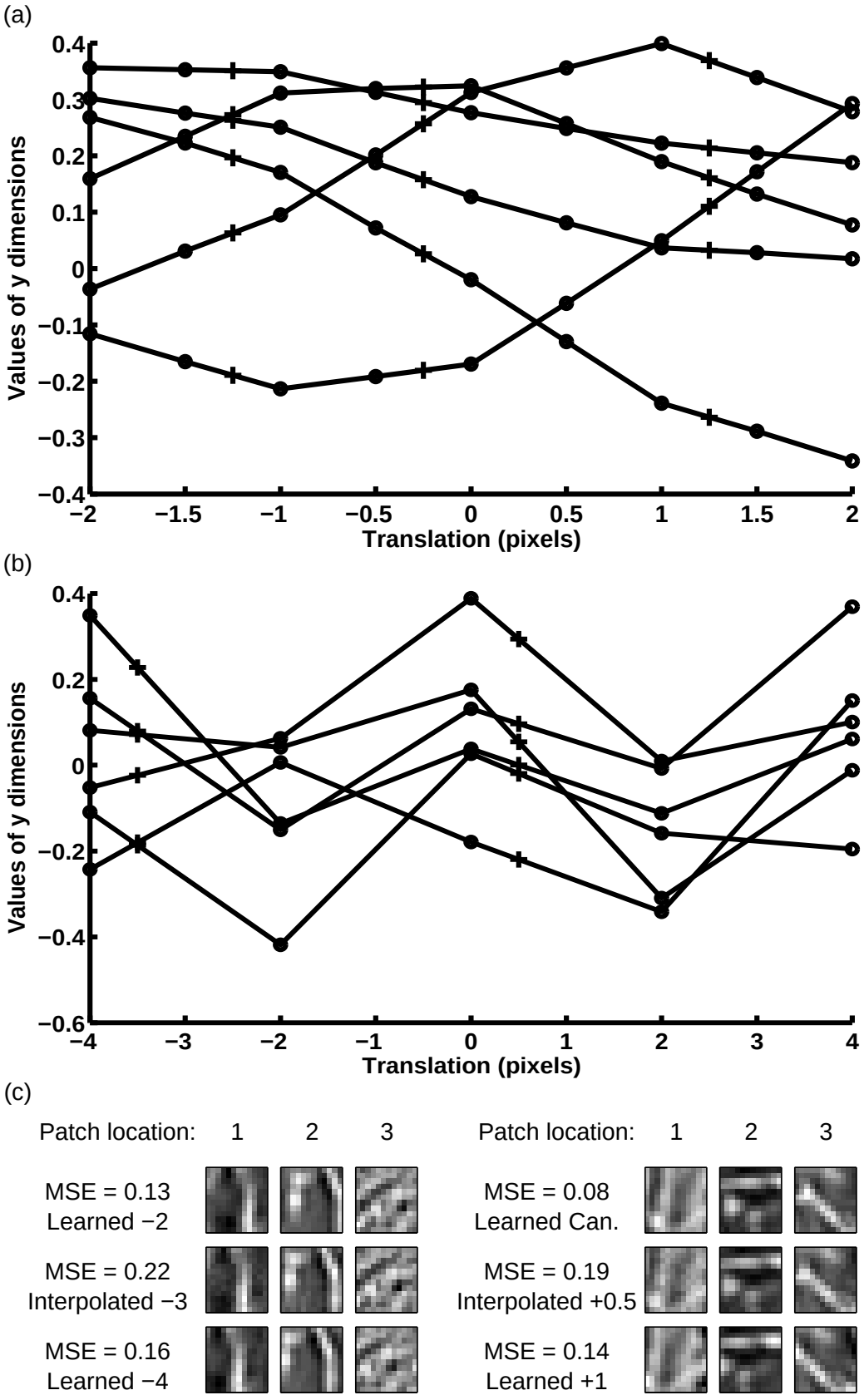

Figure 13. Interpolating in the style space allows for handling of continuous transformations. See section 4.5 for details.

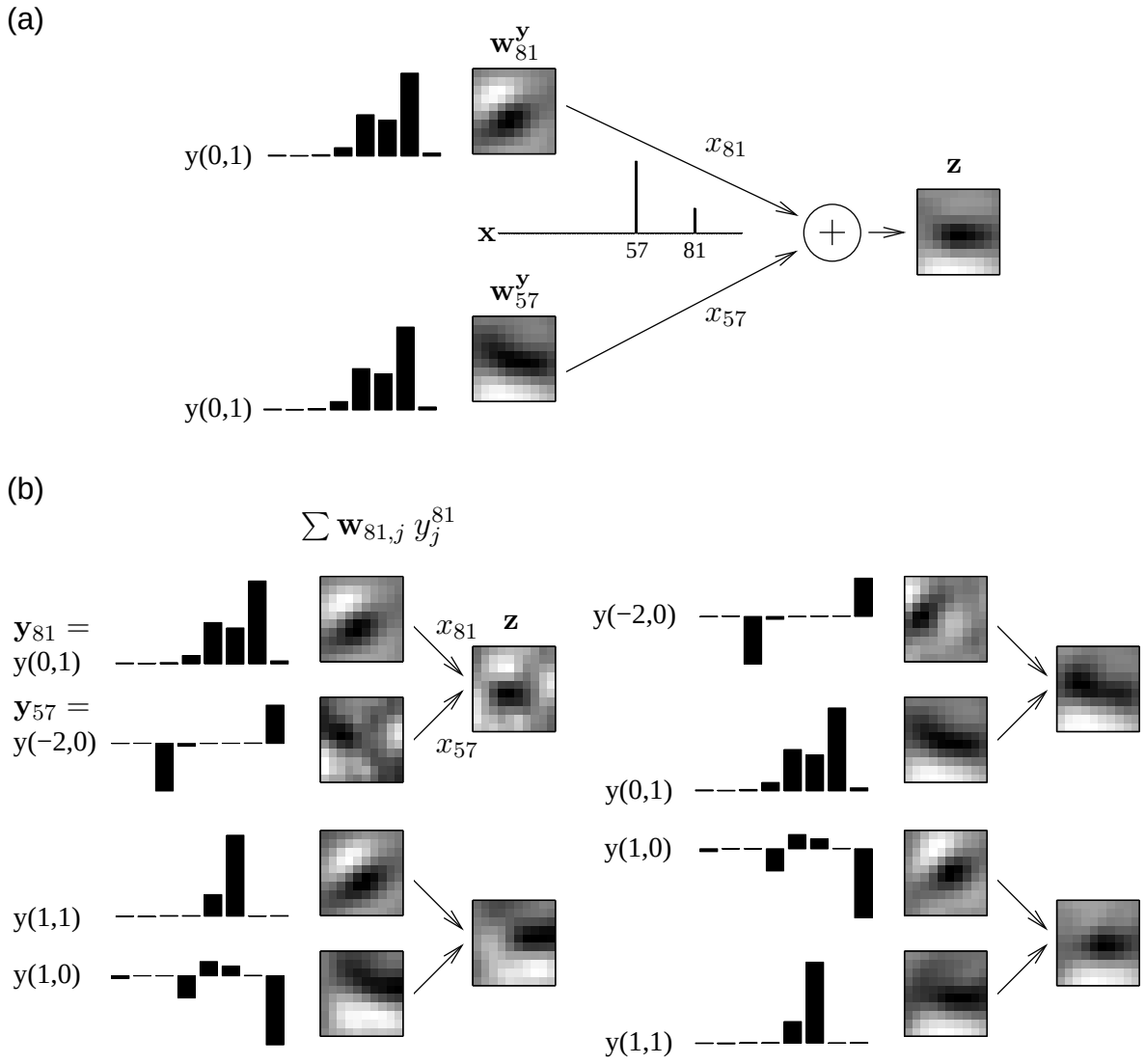

Figure 14. Modeling independently transformed features. (a) shows the standard bilinear method of generating a translated feature by combining basis vectors $\mathbf{w}_{ij}$ using the same set of y_j values for two different features ($i = 57$ and 81). (b) shows four examples of images generated by allowing different values of y_j for the two different features. Note the significant differences between the resulting images, which cannot be obtained using the standard bilinear model.

on image coding has stressed the importance of localized, independent features derived from metrics that emphasize the higher-order statistics of inputs [6, 3, 7, 5]. This paper introduces a new probabilistic framework for learning bilinear generative models based on the idea of sparse coding.

Our results demonstrate that bilinear sparse coding of natural images produces localized oriented basis vectors that can simultaneously represent features in an image

and their transformation. We showed how the learned generative model can be used to translate a basis vector to different locations, thereby reducing the need to learn the same basis vector at multiple locations as in traditional sparse coding methods. We also proposed an extension of the bilinear model that allows each feature to be transformed independently of other features. Our preliminary results suggest that such an approach could provide a flexible platform for adaptive parts-based object recognition, wherein objects are described by a set of independent, shared parts and their transformations. The importance of parts-based methods has long been recognized in object recognition in view of their ability to handle a combinatorially large number of objects by combining parts and their transformations. Few methods, if any, exist for learning representations of object parts and their transformations directly from images. Our ongoing efforts are therefore focused on deriving efficient algorithms for parts-based object recognition based on the combination of bilinear models and sparse coding.

Acknowledgments This research is supported by NSF grant no. 133592 and a Sloan Research Fellowship to RPNR.

References

[1] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3):183–193, 1954.

[2] H. B. Barlow. Possible principles underlying the transformation of sensory messages. In W. A. Rosenblith, editor, *Sensory Communication*, pages 217–234. Cambridge, MA: MIT Press, 1961.

[3] A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[4] G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions Royal Society B*, 352(1177–1190), 1997.

[5] M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, 2000.

[6] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[7] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:33113325, 1997.

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

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

[10] R. P. N. Rao and D. L. Ruderman. Learning Lie groups for invariant visual perception. In *Advances in Neural Information Processing Systems 11*, pages 810–816. Cambridge, MA: MIT Press, 1999.

[11] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[12] O. Schwartz and E. P. Simoncelli. Natural signal statistics and sensory gain control. *Nature Neuroscience*, 4(8):819–825, August 2001.

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