Learned Factorization Models to Explain Variability in Natural Image Sequences

Benjamin Jackson Culpepper

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2011-61
http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-61.html

May 13, 2011

Learned Factorization Models
to Explain Variability
in Natural Image Sequences

by

Benjamin Jackson Culpepper

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Bruno A. Olshausen, co-Chair
Professor Jitendra Malik, co-Chair
Professor Michael R. DeWeese
Professor Trevor Darrell

Spring 2011

The dissertation of Benjamin Jackson Culpepper, titled Learned Factorization Models to Explain Variability in Natural Image Sequences, is approved:

Professor Bruno A. Olshausen, co-Chair Date

Professor Jitendra Malik, co-Chair Date

Professor Michael R. DeWeese Date

Professor Trevor Darrell Date

University of California, Berkeley

Learned Factorization Models
to Explain Variability
in Natural Image Sequences

by

Benjamin Jackson Culpepper

Abstract

Learned Factorization Models
to Explain Variability
in Natural Image Sequences
by
Benjamin Jackson Culpepper
Doctor of Philosophy in Computer Science
University of California, Berkeley
Professor Bruno A. Olshausen, co-Chair

Robust object recognition requires computational mechanisms that compensate for variability in the appearance of objects under natural viewing conditions. Yet, these have proven to be difficult to engineer. For this reason, the development of computational models that achieve invariance to the types of transformations that occur during natural viewing will both benefit our understanding of biological systems and help to achieve the goals of computer vision. This thesis develops a set of models that learn low dimensional representations of the transformations occurring in dynamic natural scenes. Good models of these transformations allow their effect to be compensated through an inference process, which jointly estimates a stable percept and a parsimonious description of its appearance.

I propose a series of models based on the idea of factoring apart image sequences into two types of latent variables: a stable percept, and a low dimensional time-varying representation of its transformation. Such a two component model is a general mechanism for teasing apart the causes that conspire to produce a time-varying image. First, I show that when both components are represented by linear expansions, the resulting bilinear model can achieve some degree of image stabilization by utilizing the transformation model to explain the translation motions that occur in a small window of a movie. Yet, the recovered latent factors exhibit dependencies that motivate the investigation of a richer, exponential map as a second model for the dynamics of appearance. In addition to the translation motions captured by the linear appearance model, this richer model learns transformations that can compensate for rotations, expansions, and complex distortions in the data. Lastly, I propose a hierarchical model that describes images in terms of a hierarchy of grouped lower-level features; learning parameters in this hierarchy is enabled by a procedure that maintains uncertainty in the posterior distributions over the latent variables.

The contribution of this work is a demonstration of an adaptive mechanism that can automatically learn transformations in a structured model, which enables sources of variability to be factored out by inverting it. This is an important step, because sources of variability are the main factor causing difficulties in artificial object recognition systems, and visual invariance is also closely related to the idea of generalization, an ability that is commonly equated with intelligence. Thus, to the extent that we are able to build seeing machines that can automatically compensate for category-level variability we will have achieved some part of the goal of artificial intelligence.

Acknowledgements

I have been fortunate to work in close proximity to many extraordinary individuals who have helped me a great deal. Bruno has been an outstanding role model and advisor, and I cannot imagine being provided with a better environment to do science. His patience, friendliness, understated confidence, boundless optimism, and constant encouragement are rare qualities in a person that has such a breadth of knowledge, and has also contributed so much to science. It has been a privilege to interact with the members of my committee. I have benefitted from them not only in receiving their feedback about my own work, but also from hearing their opinions during group meetings, seminars, and hallway conversations. When Jitendra speaks, he is incredibly articulate, able to precisely express his insights on every topic of importance, and I feel fortunate for the times I have been in his company and he has shared them. Mike's boundless curiosity has been a huge asset to me, as his polite, well-mannered, but prolific questions always increase the amount of information transferred to me during scientific talks we both attend. Trevor's keen ability to get straight to the heart of any matter I discuss with him has been a remarkable resource to me, and his calm consideration and reflection always make discussions with him a pleasure.

I would be quite lost were it not for the other faculty, post-docs, graduate students and researchers around campus. My dear friends from across campus, in the Redwood Center, and especially those who sat in the same room with me, made the time pass too quickly. A large part of my education is due to these people: Charles Cadieu, Pierre Garrigues, Jascha Sohl-Dickstein, Jimmy Wang, Amir Khosrowshahi, Kilian Koepsell, Arel Cordero, David Warland, Tony Bell, Nicol Harper, Peter Battaglino, Paul Ivanov, Chris Rozell, Fritz Sommer, Daniel Little, Tim Blanche, Nicole Carlson, Badr Albanna, Pentti Kanerva, Vivienne Ming, Chris Hillar, Thomas Lauritzen, and Ivana Tosic, my wonderful, supportive girlfriend. I have also learned quite a bit from Jeff Hawkins, without whom the Redwood Center would not exist. I deeply appreciate his successful effort to set right what he found lacking about UC Berkeley during his own attendance.

The support of my family has played an extremely important role in my education, starting with my loving parents who have held it in such high regard. My sister has also been a constant source of encouragement, as have many of my aunts, uncles, cousins, and close family friends. I am indebted to Larry Leifer who let me participate in his engineering class at Stanford, and sent me over to NASA ARC to work on a joint project. Bob Keller, my advisor at Harvey Mudd College, was also a powerful force contributing to my continued interest in science, and it was from him that I first learned about and became fascinated with artificial neural networks and ICA. Finally, I thank Jeff Johnson for inquiring as to whether I had asked to be Bruno's student, and when I said no, facilitating one of the best decisions I've ever made with the convincing look on his face when he replied, 'you should!'

Contents

Chapter 1

Introduction

Despite the fact that most animals solve it naturally, vision is a computationally challenging problem that has defied efforts to capture it in algorithms. Paradoxically, the difficulty of vision comes as a shock to those unfamiliar with the subject. Because we are born with an algorithm that provides a nearly perfect solution, we tend to take it for granted. Yet, vision algorithms must determine the causal structure of the world in the presence of significant obstacles, such as the vast loss of information that occurs when projecting a 3d scene onto a pair of 2d images, and the fact that every object we recognize gives rise to an infinite number of distinct patterns on the retina. These truths, combined with evidence that a satisfactory solution has eluded many great minds, make less surprising the large amount of real estate occupied in the cortex by vision-related neural circuits, and raise a profound question. What do we compute from the retinal image that endows us with such a powerful ability to generalize and identify categories with vast amounts of variability?

1.1 What computations are required to 'see'?

"Developing general-purpose computer vision systems has proved surprisingly difficult and complex. This has been particularly frustrating for vision researchers, who daily experience the apparent ease and spontaneity of human perception [Barrow and Tenenbaum, 1981]." In the 30 years since Barrow & Tenenbaum expressed this sentiment, vision has been dissected and parceled out into a variety of problems and sub-problems. For some time we have known that vision is an under-determined inverse problem and would be impossible to solve were there not some basic constraints that apply universally: there are only three dimensions in the world; the sun is usually the primary source of light; the distance of familiar objects can be judged by their apparent size; etc. However, when a particular sub-problem in vision is considered, it immediately becomes clear that these basic constraints are insufficient, and a search for others begins. This activity has resulted in many useful facts and algorithms.

It has enabled us to identify the limitations and requirements of a particular approach, and given us an understanding of why it works, how well it works, and what its failings are. It is through this process of exploration that we know the estimation of optical flow requires a pooling of spatial constraints [Horn and Schunk, 1981], that depth discontinuities are a heavy tailed phenomena [Black and Anandan, 1996], and that precise feature matching can provide complementary information to smoothness [Brox and Malik, 2011]. However, we can only debate about whether these techniques will play a final role in a robust perceptual algorithm. It is not clear that combining the solutions to optical flow and other sub-problems will together amount to a significant whole in general purpose vision. Indeed, the problem of unifying what we know in a common framework is a difficult challenge itself, and it is not clear how to best approach it.

Our near-perfect ability to recognize categories of objects with large in-category variability under highly variable viewing conditions produced by shadows, occlusions, deformations, and positioning of parts or limbs is not well-understood. Although we have many algorithms for solving sub-problems in computer vision, there are few principles by which different algorithmic solutions to the same sub-problem can be related in a common framework. Although the biological solution to vision is certainly not the only valid one, at some level it seems relevant to ask, "Can biology be combining so many separate solutions, or are there some generic computations that can be applied to many sub-problems?" This question was answered to some extent, by [Koenderink and van Doorn, 1997], who showed that many problems in vision, including the estimation of albedo, edge detection, photometric stereo, photogrammetry, and various forms of shape from motion can be cast as inference problems in the context of bilinear models. This general framework may help to understand isolated successes of computer vision. Herein, I take an approach of this nature and focus on building representations of the sensory signals that make many visual tasks simpler. One way to achieve such a representation is to specify its desirable high level attributes in an optimization framework. These representational goals are then pursued jointly. The main goals investigated in this work are perceptual stability – the formation of a stable percept from highly variable and dynamic images – and the separation of causal latent variables.

1.2 Variability

The appearance of an object can be mainly attributed to four factors: shape, reflectance properties, pose, and illumination. There is a class of rigid objects for which shape and reflectance are intrinsic properties that do not vary. However, even this is not universally true. Many natural objects such as plants and animals have surfaces defined by deformable contours and articulated branches or limbs that move and can take on a variety of configurations. Furthermore, the reflectance properties of surfaces can change under certain conditions; for example, when they come into contact with moisture, or are polished, corroded, or abraded.

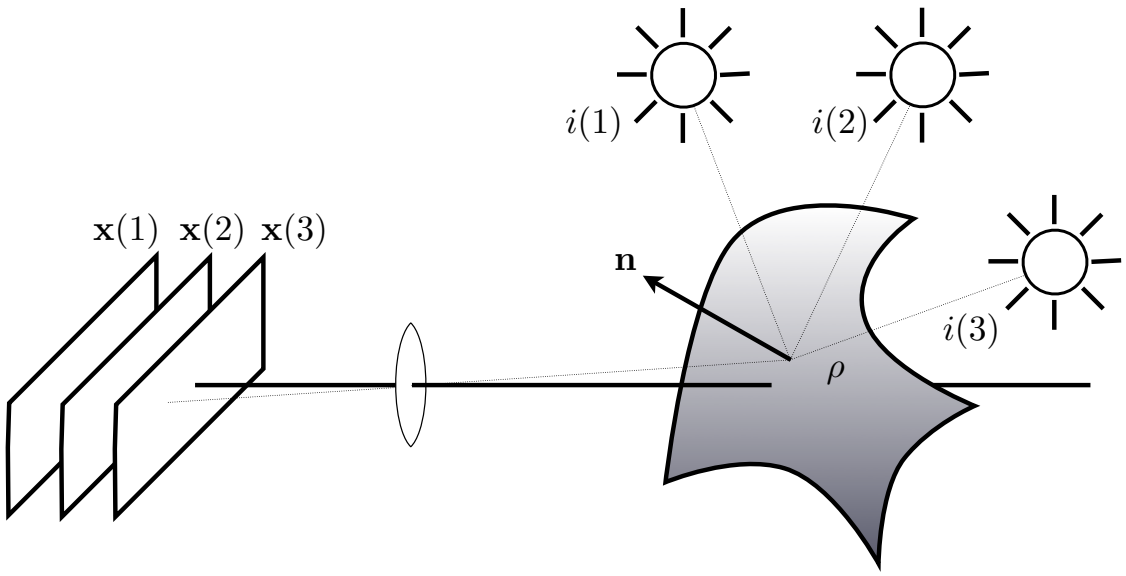


Figure 1.1: **Shape from illumination variability.** A sequence of images $\mathbf{x}(t)$ in which the illuminant $i(t)$ changes position enables the estimation of surface normals $\mathbf{n}$ and reflectance properties ρ at observed locations on the object.

In outdoor environments, the sun is the primary, and usually the only source of light, but complex shadows can easily arise from the interaction of multiple objects in a scene. Even in isolation, any one of these factors can generate a set of object appearances too numerous to count, and jointly these sources of variability render simple brute-force comparison algorithms useless for recognition purposes.

However, image variability is constrained by the physics of light. The variability produced by the cross product of shape, pose, and lighting makes recognition seem like a challenging task, but it is possible to build detailed models of how 3d shape and lighting interact to produce appearance. These models can be inverted to recover shape from appearance [Woodham, 1980], as depicted in Figure 1.1, and are largely responsible for the recent development of high accuracy face recognition systems. By having people sit still in lighting domes, their faces can be imaged repeatedly, under many different poses, illuminations, and expressions. Then, detailed 3d models of an individual's face can be constructed and used to compute the appearance of the face under an arbitrary illuminant. Although the number of pixels required to represent a face is not insignificant, with this ability to densely sample the continuous surface of face images, a clever comparison algorithm can accurately and efficiently narrow the set of candidates with which to associate a face. Although technically possible, this idea is far more difficult to apply to arbitrary objects in unconstrained environments, such as the zebra depicted in Figure 1.2.

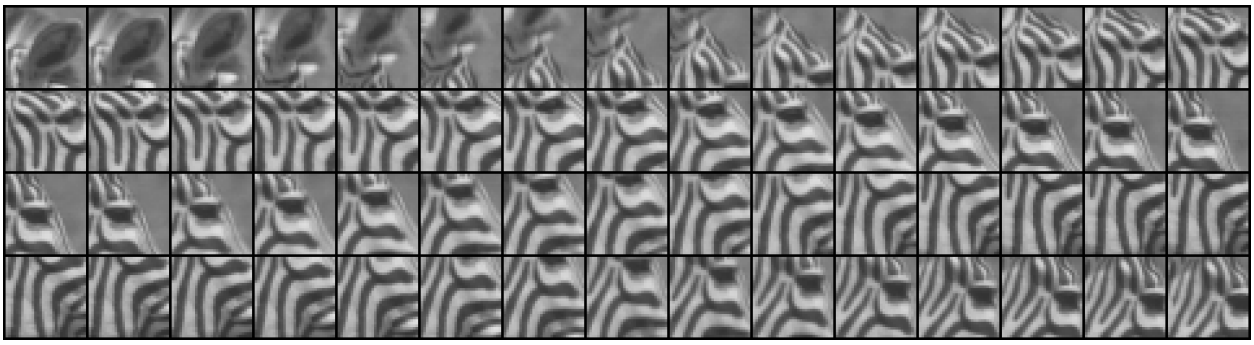


Figure 1.2: **Image variability in unconstrained environments.** 60 frames of a movie, shown through a 31x31 pixel patch (time advances left to right, top to bottom). Although our experience is a stable percept of a zebra moving its head naturally, the actual pixels vary dramatically, even between pairs of sequential frames.

1.3 Invariance

Mammalian vision systems contain machinery composed in a hierarchy which gives rise to units that are both increasingly selective to their inputs and invariant in their response properties to certain naturally occurring input transformations. We know that the perceptual machinery in animal vision contains complex cells in primary visual cortex (V1) that are selective for oriented bars, but invariant to slight displacements of the bar perpendicular to its orientation [Hubel and Wiesel, 1962]. The discovery of this fact marked an important advancement in our understanding of the cortex, and, in particular, the invariant response properties of cells in the early visual pathway. Since then, methods and approaches for forming invariant representations have been the subject of extensive study in both computer vision and neural computation, yet still relatively little is known about the mechanisms by which cortex achieves it. Fukushima's neocognitron, in which layers of features are computed and pooled to shunt variability in position, is widely regarded as the seed from which many functioning computer vision models of invariance sprouted [Fukushima, 1980]. One significant extension, Le Cun's convolutional neural network, uses the backpropagation algorithm to train layered networks in which each subsequent layer provides a progressively increasing level of spatial invariance and invariance to typical distortions of handwritten digits, and is convincing both as a model of cortical function and as a means of solving practical computer vision problems [LeCun *et al.*, 1989]. Essentially the same model has also been used to differentiate between 3-dimensional object classes, even with significant variety between object instances [LeCun *et al.*, 2004]. The idea of making comparisons stage-wise, with a hierarchy of progressive, increasingly accurate and computationally intensive modules is also widely regarded as a good design principle in both biological and artificial

vision systems [Simard *et al.*, 2001]. However, these models achieve invariance by discarding information about the signal – once discarded, this information is gone forever, since it cannot be recovered from the resulting representation.

Notably, this same behavior is exhibited in many computer vision models that have been used successfully in scene categorization and object recognition – texton models [Renninger and Malik, 2004], other 'bags of keypoints' models [Csurka *et al.*, 2004], and other models based on the neocognitron. While it may be true that certain kinds of information are useless once separated – and can thus safely be discarded – this is certainly not the case for information about spatial position and scale, and many 'bag' models do exactly this. It is only acceptable when viewed in the highly unnatural context of doing object recognition independent of any other task such as grasping. Even in this case, only coarse category information can be obtained this way, and only for certain categories. This is not to say that perception requires all information to be represented precisely as it is passed up the processing hierarchy; on the contrary: perceptual systems likely learn approximate solutions rather than explicitly describe physics or geometry. However, finding the right approximation that preserves all information of importance requires careful function design, or learning.

An alternative mechanism by which invariance can be achieved is to directly model the process by which the image of an object is transformed [Frey and Jojic, 1999; Miller *et al.*, 2000; Memisevic and Hinton, 2007]. By then inverting the forward model, the invariant properties of the object can be separated from the transformations it undergoes due to lighting, motions, changes in viewpoints, and other factors in the scene [Olshausen *et al.*, 1993; Rao and Ruderman, 1999; Tenenbaum and Freeman, 2000; Arathorn, 2002]. Since we know that many of these properties combine multiplicatively, their separation can be achieved by *factoring* the image. Even if the relationship is not purely multiplicative, multiplication is often a useful tool in obtaining a stage-wise approximation to the true relationship. In this case multiplication acts as a convenient, simple non-linearity.

Needless to say, a suitably complex task is required to measure the benefit of preserving information about transformations. In particular, object recognition in isolation, without a subsequent grasping task, may not be a suitable benchmark. However, factorization models capture many vision problems in a natural way, allowing us to think about them in a common framework. The close relationship between factorization and invariance can be viewed in the following way: invariance is an important side-effect of a proper factorization!

1.4 Factorization

Many problems in vision boil down to the factorization of two variables that interact multiplicatively. A canonical example of this is the problem of estimating the albedo of some material (or, equivalently, the radiance of the light source), given only the irradiance at its surface. In this problem the irradiance, o, is produced by a multiplicative interaction

between the light source, i, and the albedo, ρ:

$$o = \rho\, i\,. \tag{1.1}$$

Only o is observed, so without prior information the 'fair' solution, which loads both factors equally, is

$$\rho = \sqrt{o}\, a \tag{1.2}$$
$$i = a^{-1}\sqrt{o}\,, \tag{1.3}$$

where there is an ambiguous scale factor, a. To resolve this ambiguity, we must introduce constraints. Some constraints come from the way we have formulated the problem: we know the albedo must be positive and less than unity, and the radiance must be positive. By learning a parameterized model of the distribution of albedos in the environment, we can constrain the solution further. We are by no means guaranteed to eliminate the ambiguity entirely, but because data collected from the natural world is so intricately structured, we have a reasonable chance of getting ρ and i mostly right most of the time.

It is important to observe that a factoring problem cannot be solved one variable at a time. The answer obtained for one factor influences the other, and the 'invariant representation' (in this case, the albedo of the material) is only one factor. Absent any sources of uncertainly, one factor determines the other exactly. More generally, especially when we are working with phenomena for which we do not have a perfect forward model, or lack all the necessary information to invert our model, constraints can be introduced in the form of prior probability distributions on latent variables. The introduction of evidence in combination with these priors allows us to compute posterior distributions over our latent variables, which become delta functions in the limit, where our forward model is perfect and we have sufficient evidence to invert it. Uncertainty or not, because the constraints can interact in potentially subtle ways during the computation of the posterior, an unavoidable explaining away process is necessitated. The focus of my research is essentially an elaboration on this theme applied to general image transformations.

1.4.1 Bilinear models

A bilinear model factors data, $\mathbf{x}$, into two multiplicatively combined latent variables, $\mathbf{y}$ and $\mathbf{z}$:

$$x_i = \sum_{jk} \Gamma_{ijk}\, y_j\, z_k\,. \tag{1.4}$$

To familiarize readers with this equation, it is useful to express it in two other, equivalent forms, to see how $\mathbf{y}$ and $\mathbf{z}$ interact with Γ_{ijk}. First, each scalar x_i is equal to the vector-matrix-vector product $\mathbf{y}^{\mathrm{T}}\, \mathbf{\Gamma}_i\, \mathbf{z}$. In this form, it is clear that each matrix $\mathbf{\Gamma}_i$, obtained by taking

a slice of the full parameter tensor, defines an inverse covariance for a squared Mahalanobis distance between vectors $\mathbf{y}$ and $\mathbf{z}$. Alternatively, we can view the settings of $\mathbf{y}$ as linearly blending together J matrices to form a basis for $\mathbf{x}$, and the coordinates of $\mathbf{z}$ are the coefficients for that basis. In this case, we have $\mathbf{x} = (\sum_j \mathbf{\Gamma}_j \, y_j) \, \mathbf{z}$. Models of this form are called 'bilinear' because when one latent factor is held constant, a linear relationship holds between the data and the other factor. Perhaps the simplest non-linear model, it provides an elegant, utilitarian framework for achieving an explicit separation of 'what' and 'where.' Koenderink & van Doorn pointed out that many problems in vision have this particular form, and there is similar support in theoretical neuroscience literature for bilinear models being the appropriate formalism to explain spatial attention modulation [Olshausen *et al.*, 1993] and the use of multiplication to perform motion estimation [Gabbiani *et al.*, 2002]. Hinton proposed using 3-way interactions as a way to factor out viewpoint variability in neural networks receiving visual inputs [Hinton, 1981], and although the idea had a clear bilinear form, this was not formalized in mathematics until four years later [Hinton and Lang, 1985]. Olshausen's model of visual attention [Olshausen *et al.*, 1993] is bilinear in input pixels $\mathbf{y}$ and control neurons $\mathbf{c}$, which specify a route to higher level variables $\mathbf{x}$ via a linear operator $\mathbf{A}(\mathbf{c})$.

$$\mathbf{x} = \sum_{jk} \mathbf{w}_{jk} \, c_k \, y_j = \mathbf{A}(\mathbf{c}) \, \mathbf{y} \, . \tag{1.5}$$

With many control neurons, the types of routing operations that can be effected by $\mathbf{A}(\mathbf{c})$ are quite flexible, encompassing translation, scaling (with subsampling and/or interpolation), and rotation. The idea is that for the appropriate choice of $\mathbf{c}$, the transformation imposed by the scene can be compensated for, and a canonical representation routed for matching to a previously stored memory. In the case of an input that has been affected by some translation, correctly recovering the $\mathbf{c}$ variables clearly separates out 'where' information, leaving a canonical 'what' representation in $\mathbf{x}$ that can easily be matched to a memory. There is experimental evidence supporting this idea of two separate processing streams in cortex from lesion studies in monkeys, where damage to dorsal or ventral cortical regions impairs abilities of spatial perception or object identification, respectively [Ungerleider and Mishkin, 1982]. However, little is known about the mechanisms by which these streams are formed in cortex. Specifically, it is not known whether they are produced by a factorization process or two separate analysis circuits: all of the available evidence is consistent with both of these explanations.

1.4.2 Separating structure and motion from image streams

An interesting application of a bilinear model to a vision problem is in the factorization of image streams into structure and motion; in particular, the work of [Tomasi and Kanade, 1992]. Rather than operate on pixels directly, they formulate their model as the generation

of a time evolution of image keypoint coordinates, where a stationary 3-d object is observed by a moving camera. Using the singular value decomposition of a matrix, they recover the 3-d keypoint coordinates and the camera rotation (camera translation is factored out in their problem formulation), under orthography. The observed variables are P 2-d keypoint coordinates tracked through F frames. These coordinates are stacked together to form a $2F \times P$ measurement matrix $\mathbf{W}$. If the image coordinates are measured with respect to their centroid, under the assumption of no noise, $\mathbf{W}$ has rank three, and can be factored into the product of two smaller matrices:

$$\mathbf{W} = \mathbf{R}\,\mathbf{S}\,. \tag{1.6}$$

Here $\mathbf{R}$ is a $2F \times 3$ matrix describing the camera rotation at each time point, and $\mathbf{S}$ is a $3 \times P$ matrix that gives structural information in the form of 3-space coordinates of each point in a coordinate system relative to the object centroid. When the measurements are corrupted by Gaussian noise, the rank of $\mathbf{W}$ will be greater than three. However, the singular value decomposition $\mathbf{W} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^{\mathrm{T}}$ neatly factors $\mathbf{W}$ such that $\mathbf{R}$ and $\mathbf{S}$ can be recovered.

A number of coinciding factors make this solution particularly elegant and relevant. First, it shows that the separation of structure and motion, which is a plausible goal for both natural and artificial vision systems, can be formulated as structured bilinear model, and solved via factorization. Second, without introducing a few important constraints, factorization alone gives a class of equivalent, ambiguous solutions. For the case of no measurement noise, the constraints reduce the class of ambiguous solutions to just one. Lastly, even with noisy measurements, the probabilistic interpretation of the model still leaves us with a single, best solution, since the posterior is Gaussian and therefore convex. Can we somehow adapt this model to deal with more natural, non-rigid objects, and operate without the assumption of orthography and the fortuitous availability of P points tracked through F frames?

1.5 Learning to factor

There is a history in the artificial intelligence community of attempting to write down, from first principles, sets of rules that describe how to shunt or nullify sources of variability, leaving the identification of the object or part intact. Such a strategy has been taken in vision, with some success. For certain sources of variability, we *are* able to write down a set of rules that discount them. David Lowe's SIFT descriptor is an excellent example of how a considerable number of these rules about images can be integrated together, tuned manually, but relentlessly, and levered to obtain a reliable image feature description that nullifies viewpoint and lighting variability to a useful extent [Lowe, 1999]. The 'standard' instance recognition pipeline, in which stable features are extracted from images, described in a way that shunts variability, and matched against examples stored in a database is an example of both the success and limitations of such schemes [Lowe, 2004]. In an impoverished world

with only rigid, textured objects, where identification is the only goal with no required ability to cope with objects for which no model is known, this system is a viable solution. However, in the natural world, vision is not just about identification, texture is often missing, and object categories have infinite variety that can never be fully experienced a-priori. Therefore, to be useful in unconstrained environments, vision systems must have a powerful ability to generalize, and the appropriate dimensions upon which to generalize depend on the statistics of the world. These facts seem to necessitate learning.

We would like to compute a representation that factors out variability, and we know that at least some of the sources of variability in image sequences combine multiplicatively. Thus, it makes sense to recover factors in the structure of a bilinear model, which we will use as an approximation to the true forward model that forms the images we observe. What we are missing is the precise setting of parameters that will give our structured model the abilities we desire, so we formulate an optimization process that will match them to the structure of the world. Due to this automatic parameter adjustment, the symbolic meaning of the latent variables in our bilinear model may not be as clear as they are in the matrices in the Tomasi & Kanade model, and for this reason we must search for a sensical interpretation of them; that is, we must experiment with the model to understand what is has learned.

1.5.1 Separating style and content

An important representative example of a case where the parameters in a factorization model were not known in advance, but learned from data, appears in a model with nearly identical structure to that of Tomasi & Kanade. Applied to a more general class of problems, dubbed "style and content separation" problems, such as identifying a font or handwriting style across letters, or recognizing a familiar face or object seen under unfamiliar viewing conditions, the model and optimization strategy was introduced by [Tenenbaum and Freeman, 2000]. Such problems of generalization, or of untangling an observation into its underlying factors, eluded capture in a widely applicable, computationally tractable framework for many years. Indeed, Hofstadter wondered if understanding this ability of the cortex would unlock a cascade of secrets, proposing that the central question in AI might be: "What is the letter 'a'?" [Hofstadter, 1985]. Tenenbaum & Freeman address this question in their learning algorithm, which can factor apart the 'style' of a font from the 'content' of the letter.

In their work, the goal of recovering variables with a precise symbolic meaning is foregone, and replaced by the flexible notion that one set of variables (style) will change during generalization across a particular category, and another set of variables (content) will represent information about the category itself. Style s and content c are represented with parameter vectors $\mathbf{a}^s$ and $\mathbf{b}^c$, respectively. $\mathbf{y}^{sc}$ denotes an observation vector that is a bilinear function of $\mathbf{a}^s$ and $\mathbf{b}^c$:

$$\mathbf{y}^{sc} = \sum_{ij} \mathbf{w}_{ij}\, a_i^s\, b_j^c\,. \tag{1.7}$$

In this model, summing over i produces a set of basis vectors for reconstructing the input vector under the style prescribed by $\mathbf{a}^s$. Tenenbaum & Freeman call this their 'symmetric' model, giving styles and contents the same effective number of parameters. However, recognizing that sometimes linear combinations of a few basis styles may not describe new styles well, they introduce an 'asymmetric' model under which the $\mathbf{w}_{ij}$ themselves vary with style:

$$\mathbf{y}^{sc} = \sum_{ij} \mathbf{w}_{ij}^s \, a_i^s \, b_j^c \, . \tag{1.8}$$

Absent any obvious choices for constraints or prior information that could be introduced to the model, Tenenbaum & Freeman use the straightforward SVD solution. In this case, SVD can be viewed as just a fast algorithm for minimizing the following energy,

$$E = \sum_{tsc} h^{sc}(t) \, ||\mathbf{y}(t) - \sum_{ij} \mathbf{w}_{ij}^s \, a_i^s(t) \, b_j^c(t)||_2^2 \, , \tag{1.9}$$

with multiple observations for each pairing of content and style indexed by t, and $h^{sc}(t)$ is an indicator function that gives 1 if $\mathbf{y}(t)$ is in style s and content c, and 0 otherwise. Viewing the problem in terms of an energy function suggests a probabilistic interpretation, given by exponentiating the negative of the energy:

$$P(\mathbf{y}|\mathbf{a}^s, \mathbf{b}^c, \mathbf{w}_{ij}^s) = \frac{1}{Z} \exp(-E) \, , \tag{1.10}$$

where Z is the normalization constant. Casting the problem this way suggests that the use of Bayes' rule would be a clear path to a principled method for integrating constraints in the form of prior information into the model.

1.5.2 Probabilistic models for factoring out natural transformations

In contrast to the approaches of [Fukushima, 1980] and [LeCun *et al.*, 1989], where successive stages of feature extraction and pooling attempt to map the underlying invariant part of an image to a lower-dimensional space, several recent attempts have been made to learn invariant representations using probabilistic factorization models. Training such a model amounts to adjusting parameters in order to achieve a good prior for the data. What makes a good prior? Fortunately, this is defined precisely: the goal of learning is to maximize the likelihood of the model. Bayesian probability theory provides an elegant, natural way to incorporate prior information into a probabilistic model. This is especially important for factorization problems which are, by their nature, under-constrained. Priors act as constraints enabling non-sensical ambiguous solutions to be discounted from consideration, but the question of which constraints to use is itself an ambiguous problem without an obvious solution. Fortunately, the problem of understanding the constraints of natural images

has been studied extensively by statisticians; we take some advantage of their findings by pursuing two additional goals.

A second goal is to achieve a representation where the sources of variability are represented separately from the identity of the scene elements. For this reason, such models typically have latent variables corresponding to these separate causes, and these semantics are achieved either through some supervision in the training scheme or the introduction of a prior. In the Grimes & Rao model described below, this separation is achieved by designating which, among several training examples, is a special 'canonical' case which should be represented by the model without the aid of a transformation. Another mechanism is to impose a *perceptual stability* prior on the latent variables encoding object identity.

A third goal is to obtain a parsimonious explanation of the image in the sense of Occam's Razor; i.e., one that is somehow the simplest yet fully consistent with the facts. To achieve this goal, it is constructive to view vision as an inference problem in a directed graph: we would like to know 'what caused this image?' The causes of the image are clearly the most compact explanation of it, and should we obtain it, it will tell us much about what we need to know in order to interact in the environment captured by the image.

1.5.2.1 Sparse priors

There is growing evidence for sparse representations in the brain. The distribution of cell activities in visual cortical areas is highly kurtotic, with neurons that exhibit little activity for most inputs, but respond vigorously when stimulated with some feature they are selective to, causing a distribution of firing rates that is peaked at zero with heavy tails. Previous studies have shown that the features in 'sparse' dictionaries learned from natural images tend to be localized, oriented, and bandpass, and the population statistics of these quantities, when compared to those of receptive fields estimated in primate V1, are quantitatively similar [Olshausen and Field, 1996; van Hateren and van der Schaaf, 1998]. Originally motivated by the desire for parsimony in the image explanation, it has also been pointed out that sparsity could plausibly result from the pursuit of a maximum entropy neural code that is constrained by firing rate [Baddeley, 1996]. However it arises, it is a useful coding principle, and when used as a prior has been shown to improve likelihood in probabilistic models of images, enabling improved compression ratios, de-noising, and in-painting. In contrast, Gaussian priors encourage the code to become more distributed and to use all dictionary elements in a given image's representation. In this case, also known as PCA or factor analysis, more global features that resemble Fourier components are learned.

Sparse distributions are peaked at zero with heavier tails than a Gaussian. Since many distributions satisfy both properties, and the data-specific amount of tail 'heaviness' is often an unknown quantity, the most commonly used prior is the mean zero Laplacian distribution:

$$P(x) = \frac{\lambda}{2} \exp(-\lambda |x|) \,. \tag{1.11}$$

The reason for its use is partly due to its simplicity, and partly due to a serendipitous correspondence between the forms of the energy function defined this way and that of BPDN [Chen *et al.*, 1998] and Lasso [Tibshirani, 1996], which are well-known statistical methods for regularizing the solutions to least squares problems. Ironically, this preference for the Laplacian distribution is somewhat misplaced because it is not particularly heavy-tailed, and, unlike the generalized Pareto distribution, does not guarantee recoverability [Baraniuk *et al.*, 2010].

It is worth noting that sparse priors are most commonly used in conjunction with a specific optimization scheme for finding model parameters, frequently referred to as MAP-EM. In this scheme, the distribution over the latent variables is heavily approximated and what is normally thought of as a firing rate parameter, λ, must be re-interpreted as a trade-off between the quality of reconstruction and the aggregate (summed) absolute activity of the latent variables. For this reason, sparse priors are generally described in the form $P(x) \propto \exp(-\alpha\, S(x))$, where α is a parameter that controls the degree of sparsity, and $S(\cdot)$ is a 'sparseness function,' typically $S(x) = |x|$ or $S(x) = \log(1 + x^2)$.

Grimes & Rao achieve several of the goals from Section 1.5.2 in a bilinear sparse coding model, closely related to the Tenenbaum & Freeman model, which integrates a sparse prior on the $\mathbf{a}^s$ and $\mathbf{b}^s$ variables to achieve parsimony. They apply it to sequences of images that have been transformed in a natural fashion, and run experiments where they succesfully recover from a synthetically transformed image a factored (canonical) image and the transformation it has undergone [Grimes and Rao, 2005]. The sparse prior encourages the learning of localized, oriented, bandpass functions in the $\mathbf{w}_{ij}$ tensor.

Omitting the normalizing constant, the following energy function defines a negative log posterior distribution for their model:

$$E = ||\mathbf{y} - \sum_{ij} \mathbf{w}_{ij}\, a_i\, b_j||_2^2 + \alpha \sum_i S(a_i) + \beta \sum_j S(b_j)\,, \tag{1.12}$$

where β facilitates controlling a differential level of sparsity between the 'style' (transformation) and 'content' (image) variables, and $S(x) = \log(1 + x^2)$. They use a variational E-M (MAP-EM) algorithm to learn parameters that maximize the average likelihood of the model over their data set, which they generate somewhat synthetically by applying discrete 2-dimensional translations in the range $[-4, +4]$ pixels to whitened natural images. Their result is a $\mathbf{w}_{ij}$ tensor which for a fixed 'style' (translation) reliably reproduces the [Olshausen and Field, 1996] result, giving a set of localized, oriented, bandpass functions. For a fixed 'content', each of these oriented functions translate to one of several positions as the 'style' parameters are varied.

1.5.2.2 Gated Boltzmann machines

Memisevic & Hinton offer an alternative formulation of the same problem, in the style of a restricted Boltzmann machine [Memisevic and Hinton, 2007]. Modeling 3-way interactions between variables in Boltzmann machines requires a slight twist on the traditional Boltzmann machine of [Ackley *et al.*, 1985]: the introduction of a set of 'gating' variables. Hence their model is referred to as a 'gated Boltzmann machine.' A gated Boltzmann machine is roughly equivalent to the Grimes & Rao model, except there is no sparse prior. Taking the first term in the Grimes & Rao energy function, expanding the square, and keeping only the two cross terms leads to an energy function of the following form:

$$E = -\sum_{ijk} w_{ijk}\, x_i\, y_j\, h_k\,. \tag{1.13}$$

In the above equation, $\mathbf{x}$ and $\mathbf{y}$ are vectors of pixel intensities related through a transformation encoded by the weights w_{ijk} and the latent $\mathbf{h}$ vector, called a 'gating' term, whose function can perhaps be seen more clearly by writing the negative energy as $\mathbf{x}^{\mathrm{T}} \left(\sum_k \mathbf{W}_k\, h_k\right) \mathbf{y}$. Summing over k blends slices of $\mathbf{W}_k$ to form a correlation matrix, and the goal of learning is to adjust the weights so that the average correlation of the two input patterns is maximized. This maximization is accomplished via contrastive divergence, which, similar to MAP-EM, is probabilistically motivated but makes use of simplifying approximations to make it tractible.

Aside from my own work, this is the first example I know of that applied learning in a model with three-way interactions to natural sensory data and obtained a meaningful result: they let their model 'watch TV' by setting $\mathbf{x}$ and $\mathbf{y}$ to adjacent frames of spatially whitened natural movies, and after running many iterations of contrastive divergence, the $\mathbf{W}_k$ matrices learned to become 2-d spatial derivatives, arranged in patterns corresponding to translation and rotation of the input. After learning, the conditional density $P(\mathbf{h}|\mathbf{x}, \mathbf{y})$ describes a distribution over optic flows, and the MAP estimate gives the optic flow that best describes the relationship between a pair of adjacent frames. By applying the model in a 'field,' or regular, repeating lattice, the conditional density can give a distribution over a flow field for a large image.

1.6 Learning good priors for images with sparse coding

Natural images have an underlying structure that is sparse; that is, one can learn a basis such that any given image can be described efficiently in terms of coefficients of only a small fraction of its elements [Olshausen and Field, 1996; Bell and Sejnowski, 1997]. In sparse coding, the elements of the matrix $\mathbf{\Phi}$ in the following linear generative model are adjusted such that the coefficients to describe a corpus of data follow a kurtotic, mean-zero probability

distribution with heavy tails:

$$\mathbf{z} = \mathbf{\Phi}\,\mathbf{a} + \mathbf{n}\,, \tag{1.14}$$

where $\mathbf{z} \in \mathbb{R}^L$ is a data item, $\mathbf{\Phi} \in \mathbb{R}^{L \times M}$ is the basis, $\mathbf{a} \in \mathbb{R}^M$ is a vector of coefficients, and $\mathbf{n}$ is i.i.d. Gaussian noise with variance σ^2; that is,

$$\mathbf{z}|\mathbf{a} \sim \mathcal{N}(\mathbf{\Phi}\,\mathbf{a}, \sigma^2\,\mathbf{I})\,. \tag{1.15}$$

It has been shown that when $\mathbf{\Phi}$ is fit to natural images using

$$-\log P(\mathbf{a}) = -\sum_m \log P(a_m) \propto \sum_m \log(1 + a_m^2)\,, \tag{1.16}$$

the activities of the $\mathbf{a}$ coefficients have response properties that are both qualitatively and quantitatively similar to those of simple cells in V1 [Olshausen and Field, 1996; van Hateren and van der Schaaf, 1998]. Furthermore, the columns of $\mathbf{\Phi}$ become localized, oriented, and bandpass, and therefore the set of active coefficients can be viewed as a symbolic representation of the image. Activating a single coefficient activates a group of pixels in the image, and unlike a single pixel, a symbolic meaning can be attributed to this group: it is an edge. A similar grouping emerges if the prior over the coefficients is factorial i.i.d. Laplacian:

$$-\log P(\mathbf{a}) = -\sum_m \log P(a_m) = \lambda \sum_m |a_m| - \log(\frac{\lambda}{2})\,. \tag{1.17}$$

Under this assumption, the negative log of the posterior probability of an image under the model (omitting the normalizing constant),

$$E = \frac{1}{2\sigma^2}||\mathbf{z} - \mathbf{\Phi}\,\mathbf{a}||_2^2 + \lambda\,||\mathbf{a}||_1\,, \tag{1.18}$$

corresponds to BPDN [Chen *et al.*, 1998], which is an alternative formulation of the Lasso [Tibshirani, 1996] problem. Besides the fact that these problems have been studied extensively in the statistics literature, an attractive property of this energy function is that it is convex in $\mathbf{a}$ when $\mathbf{\Phi}$ and $\mathbf{z}$ are held constant. This convexity is important, because finding $\arg\min_{\mathbf{a}} E$ is equivalent to computing the MAP estimate in the probabilistic model, $\arg\max_{\mathbf{a}} P(\mathbf{a}|\mathbf{z})$ – i.e., the most likely explanation of an image under the model. A straight-forward manner to find this solution is to use the gradient of the energy function,

$$\Delta\mathbf{a} \propto -\frac{\partial E}{\partial \mathbf{a}}\,, \tag{1.19}$$

but there are also many more efficient and accurate algorithms [Efron *et al.*, 2004; Kim *et al.*, 2007; Figueiredo *et al.*, 2007; Rozell *et al.*, 2008].

Since these definitions specify a probabilistic model, there exists a Monte Carlo algorithm to estimate the parameters $\mathbf{\Phi}$, but such algorithms are usually computationally expensive. A faster, approximate scheme for learning $\mathbf{\Phi}$ was proposed by [Olshausen and Field, 1997] and has since become quite popular and known as MAP-EM. The idea for the approximation stems from the observation that since $P(\mathbf{a})$ is sparse, then the distribution $P(\mathbf{a}|\mathbf{z})$ will be tightly peaked, and thus adequately described by a single sample taken at its maximum. In the estimation algorithm, $\mathbf{\Phi}$ is refined using an iterative, alternating scheme in which MAP estimates are computed for a few randomly selected data items using a fixed $\mathbf{\Phi}$, and then a small update to $\mathbf{\Phi}$ is taken by computing

$$\Delta\mathbf{\Phi} \propto -\frac{\partial E}{\partial \mathbf{\Phi}}\,, \tag{1.20}$$

using the MAP-estimated values for $\mathbf{a}$. Starting from a random initial $\mathbf{\Phi}$, this procedure is repeated until the fractional energy decrease from one iteration to the next falls below some threshold.

1.6.1 Capturing variability in learned group structures

The representation provided by MAP-estimated latent variables in sparse coding has been shown to improve static image classification results [Raina *et al.*, 2007; Ranzato *et al.*, 2007; Mairal *et al.*, 2008b; Mairal *et al.*, 2008a; Yang *et al.*, 2009; Boureau *et al.*, 2010; Yang *et al.*, 2010; Kavukcuoglu *et al.*, 2010]. Thus, though highly simplified, a linear generative model with a sparse prior yields a practically useful transformation of pixels that has a quantifiable relationship to the response properties of cells in cortical visual areas which we believe to be highly optimized machines for processing visual input. However, despite the fact that the representation can be used to improve performance in a classification task, it is not a structured model for capturing variability. In fact, small perturbations in the input will give rise to dramatically different MAP estimates in the $\mathbf{a}$ variables, which in some sense is exactly the opposite of what we would like.

To illustrate this point, consider the boxed portion of the whitened image shown in the left portion of Figure 1.3 – the area of the box is 16x96 pixels. Imagine a camera slowly panning down this area, foveating windows of 16x16 pixels. In the figure, the y coordinate gives a window position. As the window translates down the image, each 16x16 sub-window is coded in terms of a 256 element basis by solving

$$\arg\min_{\mathbf{a}(y)} \frac{1}{2\sigma^2}||\mathbf{x}(y) - \mathbf{\Phi}\,\mathbf{a}(y)||_2^2 + \lambda\,||\mathbf{a}||_1\,, \tag{1.21}$$

producing the 80 rows of coefficients on the right. The basis $\mathbf{\Phi}$ is learned from whitened images using sparse coding. After learning, the columns of $\mathbf{\Phi}$ are sorted in increasing order

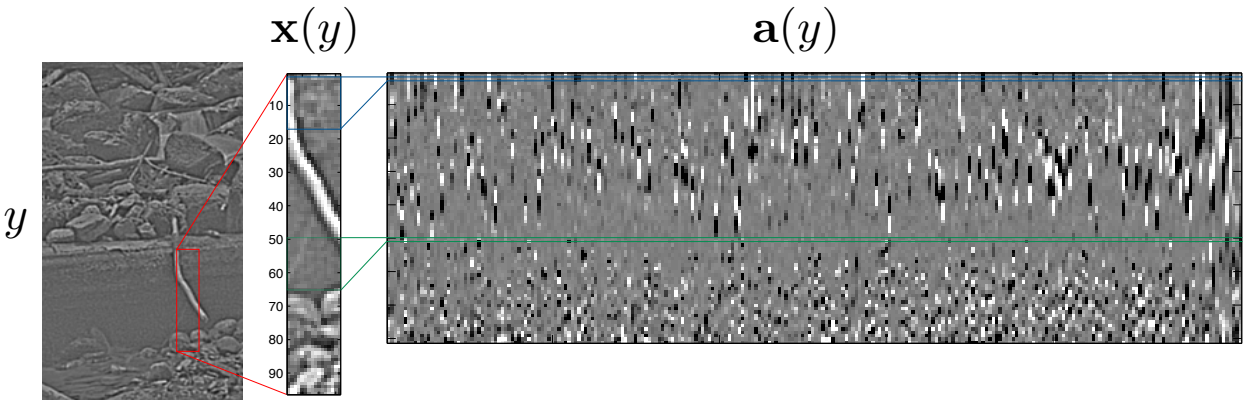


Figure 1.3: **Sparse codes and image variability.** (left) One of the most prominent statistical properties of natural images is that the intensities of spatially adjacent pixels are correlated. By first removing this effect using a simple de-correlation procedure known as whitening, more sophisticated models can focus exclusively on higher-order statistics that are more difficult to capture. On the left is shown a whitened image of a natural scene with the section boxed in red expanded to show the detail of a clear edge caused by a branch, and some textured rocks below it. (center) Consider each of the 16x16 sub-windows of the expanded vertical slice labeled $\mathbf{x}(y)$. The y axis, labeled at the far left, gives the position of the 16x16 window. At the top, $y = 1$, and the region boxed in blue is the second 16x16 sub-window, representing $\mathbf{x}(2)$. The region boxed in green is the 50-th sub-window, representing $\mathbf{x}(50)$. (right) Each of these sub-windows has been encoded by a 16x16 pixel sparse basis whose elements have been ordered by decreasing spatial frequency from left to right. That is, each row of the box labeled $\mathbf{a}(y)$ gives the solution to $\arg\min_{\mathbf{a}(y)} \frac{1}{2\sigma^2}||\mathbf{x}(y) - \mathbf{\Phi}\,\mathbf{a}(y)||_2^2 + \lambda\,||\mathbf{a}(y)||_1$. Positive values are light, and negative values are dark. The row boxed in blue corresponds to the code for $\mathbf{x}(1)$, the matching 16x16 window in the center column boxed in blue. Correspondingly, the row boxed in green corresponds to the code for $\mathbf{x}(50)$, the matching 16x16 window in the center column boxed in green. Note that for y values from 20 to 40, between the blue and green boxed regions, there is a clear edge being encoded in the window. Yet rows 20 to 40 in the code vary dramatically.

of spatial frequency. Since the columns of $\mathbf{\Phi}$ have been sorted this way, the activities of the coefficients towards the right in the columns of the middle panel in the figure change more slowly than that of those towards the left. This rate of change can be thought of as the amount of invariance the image model alone has to changes in the underlying pixels (the coefficients are certainly changing more slowly than the pixels).

Focus on rows 20 to 50 of the coefficient activities. Across these rows, the symbolic content of the window remains essentially unchanged – there is a single edge – yet, the activities of the coefficients change dramatically as the window slides. The reason dramatic coefficient changes result from slight changes in the image content is that content and position are confounded in the representation – the 'what' and the 'where' have not been separated. If our model was able to account for the dramatic change in the pixels as being caused by a motion rather than a change in content, the encoding of the content could remain fixed as the motion occurred. This is the goal we hope to achieve by stabilizing the image component of our model through the use of a time-varying transformation.

The variability exhibited by the coefficients during the window sliding experiment is dramatic, and corresponds directly to regularities in the structure of the world. Presumably, methods that capture these strong statistical dependencies and code the image in a way that is stable under normal, changing circumstances in the environment should be capable of yielding even better performance in classification tasks, and other applications. Clearly, the aforementioned model is not sufficient; therefore, an important question to ask is: what kind of model should we propose?

One approach to modeling these dependencies is based on subspace models which attempt to impose a group structure on the coefficients, where sparsity is imposed on groups rather than individual members within a group [Hyvarinen and Hoyer, 2000; Cadieu and Olshausen, 2009; Garrigues and Olshausen, 2010]. However, this approach is still unsatisfying when the group structure is prescribed rather than learned. A few recent undirected probabilistic models of static images have suceeded in learning these groupings [Köster and Hyvärinen, 2007; Ranzato *et al.*, 2007; Ranzato *et al.*, 2010a], and this research direction has been gaining interest.

Work on modeling image transformations [Memisevic and Hinton, 2007; Memisevic and Hinton, 2010], separating content and style [Freeman and Tenenbaum, 1997; Grimes and Rao, 2002], and capturing higher-order dependencies in natural images [Karklin and Lewicki, 2003; Karklin and Lewicki, 2006; Osindero *et al.*, 2006], can all be viewed as attempts to learn grouping structures. Additionally, these models all have roots in the simple but powerful bilinear model, which first drew attention from the vision community when it was used to separate an object's identity (or 'content') from its pose ('style') with some success [Freeman and Tenenbaum, 1997]. Ironically, this important success story is somewhat of a departure from the structure of a bilinear model, as identity and pose do not interact strictly in a multiplicative fashion – which raises an important question: since the model does not match the structure of the problem, why use it? Just as a linear model is often not a perfect fit to

a problem, a bilinear model may not be a perfect fit. However, it is capable of compactly representing a diversity of non-linear functions – though not so many that it suffers greatly from over-fitting. Simply put, it is the simplest non-linear model. This non-linearity makes it more powerful, but also more difficult to estimate. Yet, it is less difficult to estimate than almost every other non-linear model. Tenenbaum & Freeman's procedure to fit bilinear models – an iterative, alternating optimization method based on SVD – was not described in a probabilistic context, but if Gaussian priors are assumed then their technique can be understood as a variational (MAP) EM based algorithm [Freeman and Tenenbaum, 1997]. Similar approaches have also been investigated with sparse priors [Grimes and Rao, 2002; Karklin and Lewicki, 2003; Garrigues and Olshausen, 2007; Cadieu and Olshausen, 2009; Garrigues and Olshausen, 2010]. In some sense, learning these groups is tantamount to finding an invariant; thus, these models are highly related to the topic of this thesis.

1.7 Summary and outline of the thesis

In the following chapters, I develop adaptive models of the dynamics of appearance, which can be inverted to recover stable percepts, a computational requirement for vision. Such models are carefully designed equations relating data (pixels) to the latent causes that produce them (objects). These causes are recovered by an 'explaining away' mechanism that operates using feedback through a hierarchy of evidence. The proper recovery of these causes is of paramount importance, as it converts the difficult challenge of answering high level, abstract questions about the image into a far simpler task. While many vision researchers attempt to achieve this indirectly, possibly through a carefully engineered sequence of filtering and pooling operations, I express the goal directly, in a structured framework.

Bilinear models have an evident synergy in their dual applicability to neural models and vision problems. As they are an abstract structure that can be applied to many vision problems, theoretic advancements could potentially lead to wide-spread improvements in vision applications that need to be robust and operate in unconstrained natural environments. For this reason, I view them as an attractive subject for long-term philosophical consideration. Through the application of sufficient thought and effort, it seems likely that an elaboration on the theme of bilinear models incorporating the correct priors may bring us closer to understanding the computations of vision. In this thesis, I develop a novel probabilistic bilinear models of images and how they transform in time. Additionally, I develop algorithms to train and evaluate these models at larger scales than have previously been investigated. The models discover efficient, causal image representations with hierarchical structure, using unsupervised learning.

In Chapter 2, I describe a factorization model that uses sparse and temporal stability priors to learn the structure of transformations in natural movies. The core of the model is the same as Olshausen's routing circuit given in Equation 1.5, but it is derived from

first principles by describing a short movie sequence in terms of a linear first-order ODE. The solution to the ODE is approximated by a first order Taylor expansion. The learned transformation operators form efficient ways to tile the distribution of translations in the training data.

In Chapter 3, the first order Taylor approximation is forgone, and parameterizations of the exponential map are learned. As only two operators are required to fully represent translation motion in this model, different tiling properties emerge for the learned parameters when they are trained using natural image sequences: the operators become localized, sometimes transforming only a portion of their input, and learn to code expansions, rotations, shearing, and other non-uniform distortions as well as translation.

In Chapter 4, I anchor the ideas described in previous chapters to statistical estimation methods and quantify the quality of several models, showing a sequence of improvements.

In Chapter 5, I conclude with an overview of the main ideas presented, and some discussion of the direction I plan to take in the future.

Chapter 2

Learning bilinear image sequence transformation models

We would like to build a machine that 'sees,' as animals do; that is, computes some function of the sensory input that enables it to know *what* is *where* in the environment. A natural way to obtain these two separate symbolic representations is by fitting a model with two components to the sensory input. However, because a significant part of the grand goal of sight can be captured in a function from images to categories, determining only this function is and has been a focal point of computer vision research. Although progress has been made, current artificial vision systems are somewhat limited in terms of the types of tasks they can perform robustly in unconstrained natural environments. There are no robust algorithms for recognizing deformable objects defined by articulated contours and those without texture, and artificial vision systems make strange, un-natural mistakes, such as confusing seemingly unrelated categories.

The fundamental problem that causes difficulty for state of the art vision systems is variability, which in images comes from a variety of different sources. All visual objects are subject to lighting and viewpoint variability. However, modern computer vision systems have been somewhat successful in discounting changes in appearance due to these factors. For example, many artificial vision systems succeed by computing specific fixed visual features and sending them to classifiers in a fully feed-forward manner, demonstrating that for some level of performance on constrained tasks, feedback and learning are not necessary. Though viewpoint and lighting are by no means simplistic sources of variability, in some ways they are simpler than others, in that it is possible to write down sets of rules that describe their effect. Given a significant, but measurable, amount of configuration information – such as the exact position and orientation of the camera and light sources, their intensity, and the position and configuration of reflecting surfaces – it is possible to directly compensate for them. This is not the case for many other sources of variability: there are many sources for which we cannot write down the forward model, or we cannot measure sufficient configuration

information to properly invert it. In these cases, a system that can automatically learn rules that achieve invariance would be useful, as it would greatly expand the number of sources of variability that our artificial vision system can compensate for.

Sources of variability that are not properly compensated for are a clear obstacle for current object recognition systems. One way this compensation process can fail is if no model for the source variability is available. However, even if the source model is available, inverting it properly is another difficult challenge. In object recognition systems that focus exclusively on the 'what' pathway, considering only one part of the problem in isolation, this inversion process could actually be made more difficult – not simpler, as the limited scope might suggest. The reason is that these factors should interact with each other during an explaining away process that takes place in the model inversion. Because both latent factors can help contribute to finding a globally consistent solution, using only one would likely lead to locally but not globally consistent explanations.

Since objects in the world persist across time, movies of objects can tell us about how their visual image transforms due to natural factors in the environment. Can we learn a representation to describe these natural transformations, by passively watching movies? A model that is capable of *learning* how to *factor* sequences of images into two more stable components – one representing an object's identity, and another capturing scene elements that cause it to appear in a specific way – would be a valuable tool, from which more sophisticated artificial vision systems could be built. The development of such a model is the subject of this chapter. We proceed by first proposing a simple model of time. Subsequently, we assemble, from two component models, a factorization model that separates an image sequence into two more stable percepts: one describing an invariant underlying image, and another that describes how it is transforming in time. Importantly, transformations describing the dynamics of appearance are learned from the image sequences themselves by fitting model parameters. Our model is structured in a hierarchy with two levels of representation, and functions to transform a gray-level image into a more symbolic form through an explaining away process. The result is a representation where the main percept in the field of view is stabilized with respect to distortions and other transformations due to natural viewing circumstances.

2.1 Modeling time-varying images

Consider a continuously sampled image of the world whose values have been concatenated into a real vector $\mathbf{x} \in \mathbb{R}^L$. A simple model of time dynamics is a first-order linear differential equation:

$$\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x}\,, \tag{2.1}$$

where the time derivative of the image $\dot{\mathbf{x}}$ is related to the image itself through a linear operator $\mathbf{A} \in \mathbb{R}^{L\times L}$. The matrix $\mathbf{A}$ fully specifies the behavior of the system from any

initial condition. Such a model is attractive primarily because it is understandable and the mathematics are tractable, yet still powerful enough to perfectly capture a number of natural phenomena under Newtonian kinematics. The solution to the above system is

$$\mathbf{x}(t) = \exp(\mathbf{A}\, t)\, \mathbf{x}(0)\,, \tag{2.2}$$

where $\mathbf{x}(0)$ is an initial condition and $\exp(\cdot)$ is the matrix exponential:

$$\exp(\mathbf{A}) = \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{A}^k\,. \tag{2.3}$$

Time is advanced from the initial condition by multiplying a scalar t with the matrix $\mathbf{A}$, exponentiating, then applying the resulting linear operator to the image. Though too impoverished to capture all of the dynamic effects of moving cameras and objects in natural movies, linear ODEs represent a good footing from which to build, as they are a perfect model for several important types of global transformations that are natural and occur frequently, such as translation. The approach taken here is to locally approximate the data by a linear ODE at each time point. To make this idea explicit, denote the initial condition to be some arbitrary point in a long sequence of frames, $\mathbf{x}(t)$. Local perturbations δ around t can be generated using the following equality:

$$\mathbf{x}(t + \delta) = \exp(\mathbf{A}(t)\, \delta)\, \mathbf{x}(t)\,. \tag{2.4}$$

Whereas the use of a single matrix $\mathbf{A}$ is insufficient to capture the time dynamics of a whole movie, this model, which utilizes a different matrix at every time, $\mathbf{A}(t)$, is now so rich that there is practically no phenomena it cannot capture. Instead, what we want is a model with just enough complexity to capture the dynamics. A simple way to scale back the power of the model is to utilize a lower dimensional representation in the functional form for $\mathbf{A}(t)$, with adjustable parameters. Then, our goal is to learn good parameters of the function $\mathbf{A}(t)$ from the statistics of natural images, where 'good' means that it should allow us to easily register pairs of sequential frames in a movie. We do this by fitting the model – estimating $\mathbf{A}(t)$ – using what is essentially a brightness constancy assumption. In performing this estimation, the goal is to find a matrix that effects a proper remapping, in which every output pixel $\mathbf{x}(t+\delta)$ is derived from some combination of input pixels $\mathbf{x}(t)$. When all of the entries in $\mathbf{A}(t)$ are zero, the identity matrix copies every value of $\mathbf{x}(t)$ to $\mathbf{x}(t + \delta)$. If $\mathbf{A}(t)$ contains negative ones along the diagonal, this copying operation will be entirely negated. A translation can be effected by first installing this diagonal of negative ones, then adding a second diagonal with all ones. Expansions and contractions can be effected by mapping each input pixel to multiple output pixels, or averaging over multiple input pixels to produce a single output pixel.

This type of approach is often referred to as a *direct* method [Irani and Anandan, 1999], because we register pixels themselves, rather than a set of sparsely sampled image features that summarize them. Direct methods fell out of favor over the last decade because they use all of the available information in an image, and are more computationally intensive. However, there has been a recent resurgence of interest in them, because their sparse counterparts often discard information that improves accuracy, and modern processors have ameliorated many of the computational constraints that previously drove the investigation of sparsely sampled features. Note that sparse representations do not have to discard information, and thus it is likely possible to combine these ideas in a way that obtains computational benefits without sacrificing accuracy.

For unstructured $\mathbf{x}(t)$, it is conceivable that the distribution of matrices $\mathbf{A}(t)$ that register pairs of sequential frames would be uniform over $\mathbb{R}^{L\times L}$. However, natural image sequences are highly structured. Variability in the appearance of objects in the world must obey physical laws governing motion and the reflectance of light. For this reason, it is not unreasonable to assume that there is a high concentration of mass in this distribution, which we can take advantage of in the formulation of a model for $\mathbf{A}(t)$. Specifically, we can parameterize it using fewer than $L \times L$ latent variables. The simplest way to do this is to express $\mathbf{A}(t)$ as a linear combination of some basis:

$$\mathbf{A}(t) = \sum_j \mathbf{\Psi}_j \, c_j(t) \, . \tag{2.5}$$

In this case, the computation required to fit the model to an image sequence is reduced from the consideration of $L \times L$ variables per pair of frames to only J.

For small δ, the first two terms in the sum of Equation 2.3 serve as a reasonable approximation:

$$\exp(\mathbf{A}) \approx \mathbf{I} + \mathbf{A} \, . \tag{2.6}$$

The use of this approximation in combination with the low-dimensional parameterization of $\mathbf{A}(t)$ means that the model may now be incapable of perfectly describing the dynamics in real image sequences. Thus, we introduce a noise variable, $\mathbf{n}(t)$, to explain portions of the data that are not captured by the model:

$$\mathbf{x}(t+\delta) = (\mathbf{I} + \delta \, \mathbf{A}(t)) \, \mathbf{x}(t) + \mathbf{n}(t) \, . \tag{2.7}$$

In this section, we have introduced a simple model of time and briefly explained how it might be parameterized and fit to sequences of natural images. At this point, it is useful to divert briefly and consider two contextual questions: Why is such a model useful, and how does it relate to vision? The fundamental problem being solved by fitting the $\mathbf{c}(t)$ variables to a sequence is *correspondence*, which is widely considered to be a computational requirement for both artificial and biological vision systems. In artificial vision systems it is closely related

to optical flow, depth from stereo, and 3d structure from motion. Additionally, the process of learning suitable parameters $\mathbf{\Psi}_j$ is akin to learning a rich prior for motion. Since the introduction of a simple, smooth motion prior [Horn and Schunk, 1981], several researchers have attempted to learn more nuanced priors to avoid smoothing over regions where multiple separate motions occur. In these cases, such as regions containing transparency or occlusion boundaries, smoothing is clearly improper. An early example of one such effort that bears a close resemblance to the approach described here is the work of [Jepson and Black, 1993], who used an E-M algorithm to estimate the parameters of a Gaussian mixture model for describing multiple motions in patches of sequential images. An overview of work that followed up on this idea is given in [Roth and Black, 2007], who did a careful study of the statistical properties of optical flow fields, and showed that many of them could be captured in a Field-of-Experts style model. A similar technique was used by [Sun *et al.*, 2008] to model the joint statistics of image intensity and flow.

A handful of researchers have made use of exponential maps to model the variability caused by complex 3d motions. In perhaps the earliest example of its use in vision, exponential maps were applied to model kinematic chains that generate articulated human motions [Bregler and Malik, 1998]. The approach described here is also highly related to the exponential maps used to learn transformations between sequential movie frames by [Rao and Ruderman, 1999].

In biological vision systems, there are a variety of neural constructs that seem to participate in the computation of correspondence: Reichardt detectors in insects [Reichardt, 1969] are a particularly simple example, followed in order of increasing complexity by cells found in areas MT and MST in human, macaque, and cat. Good predictive models of Reichardt detectors combine the use of coincidence detection and specific propagation delays. To a lesser extent, we are also able to predict the firing properties of cells in MT [Zemel and Sejnowski, 1998]. In this chapter, our goal is not specifically to develop a predictive model of biology, but we hope that by optimizing it to natural image sequences it may learn a solution that has some analogue in nature. If this turns out to be true, in addition to its main purpose as a functioning part of an artificial vision system, the model proposed here may provide new insights about the types of stimulae that may effectively characterize the behavior of higher-level MST cells, which often pool outputs from multiple MT cells, and whose response properties are less well-understood.

2.2 Learning natural image transformations from movies

Up until recently, sparse coding has been mostly used to learn the *content* of natural images. Now we turn to a second problem: learning how the content of the image is transforming in time. Like static images, time-varying natural images also have an underlying structure that is sparse [Olshausen, 2003], and due to the persistence of objects through time, they have

time-invariant components [Wiskott and Sejnowski, 2002; Hyvarinen *et al.*, 2003; Einhauser *et al.*, 2002]. In this section, we describe algorithms for fitting the model to natural image sequences by learning $\mathbf{\Psi}_j$ and $\mathbf{\Phi}$.

We begin by learning the model parameters one component at a time. Since the camera captures images of the scene at a fixed interval, our measurements will occur at integer values for t for some choice of units. Additionally, if we restrict ourselves to model only frame-to-frame transformations, we can fix $\delta = 1$. We impose a Laplacian prior on the $\mathbf{c}$ variables, with parameter γ, giving the following form for the negative log of the posterior probability of an image sequence under the model (omitting the normalization constant):

$$E = \sum_t \frac{1}{2\sigma^2} ||\mathbf{x}(t+1) - (\mathbf{I} + \sum_j \mathbf{\Psi}_j \, c_j(t)) \, \mathbf{x}(t)||_2^2 + \gamma \, ||\mathbf{c}(t)||_1 \, . \tag{2.8}$$

Note the similarity to Equation 1.18; in fact, the inference problem in this case is again equivalent to BPDN. Our learning procedure for $\mathbf{\Psi}_j$ is MAP-EM, and thus quite similar to that described in Section 1.6. However, here our goal is to learn a set of basis functions $\mathbf{\Psi}_j$ from pairs of sequential frames of a natural movie such that the correspondence between any pair can be described compactly using a small number of non-zero coefficients $\mathbf{c}$.

The raw data used in these experiments comes from a BBC documentary filmed in Africa, and is exemplified in Figure 1.2. As a form of pre-processing, we perform PCA dimensionality reduction and whitening on the individual frames of our image data before using it in training (see Appendix 2.A for further details). One final point to mention is that to prevent the model from trying to transform unrelated content from frame t into new content that has entered into frame $t+1$, a binary mask that contains zeros along the boundary of the patch is multiplied point-wise with the reconstruction error, before computing its L2 norm.

2.2.1 Inference

Since we approximate the posteriors over our latent variables by samples taken at the MAP-estimate, doing inference in the model is equivalent to computing MAP-estimates. The optimization problem that must be solved to compute these estimates is again equivalent to BPDN, but converting it to this form requires precomputing the tensor $\Theta_{ljt} = (\mathbf{I} + \mathbf{\Psi}_j) \, \mathbf{x}(t)$. This operation is memory intensive, and adds computational overhead to inference, marginalizing the gains one would achieve from using a specialized BPDN solver. In the following experiments, we set $\gamma = 0.1$, and the L-BFGS algorithm [Liu and Nocedal, 1989] is used to compute $\arg\min_{\mathbf{c}(t)} E$ directly. For $J < L$, this optimization problem has a unique minimum. However, in practice the optimization is never run to completion – it is run until the fractional energy decrease between iterations of L-BFGS falls below some threshold, or a maximum number of L-BFGS iterations is exceeded. For this reason, and to be consistent with the procedure used in all the transformation models described herein, we solve this

optimization via the continuation method described in Appendix 2.B. This serves to steer the solution towards something sensible in the cases where the optimization is terminated early.

2.2.2 Learning

Learning is accomplished by adjusting the $\mathbf{\Psi}_j$ from a random initialization, according to

$$\Delta \mathbf{\Psi}_j \propto -\frac{\partial E}{\partial \mathbf{\Psi}_j}\,, \tag{2.9}$$

as described in Section 1.6. Each iteration, the inferred coefficients for a 10 frame image sequence, randomly selected from the much larger movie, is used to compute a learning gradient. After each learning step, an additional constraint on the Frobenius norm of each $\mathbf{\Psi}_j$ element is imposed:

$$||\mathbf{\Psi}_j||_{\mathrm{F}} = L \quad \forall\, j\,, \tag{2.10}$$

with forces each $\mathbf{\Psi}_j$ to have the same norm as $\mathbf{I}$. The purpose of this constraint is to guide the learning process early on, preventing the $\mathbf{\Psi}_j$ from finding idiosyncratic solutions with dramatically long norms, and from collapsing to a small norm, essentially turning off, instead of learning something useful. Figure 2.1 shows the result for $P = 225$, $L = 100$, $J = 9$, projected back to pixel space for display. These top 100 PCA components capture more than 99.12% of the variance in the data. At this small value for J, the most obvious pattern that has been learned by the basis elements is spatial locality and consistency. That is, all pixels are transformed using only pixels in a local region around them, and in each learned $\mathbf{\Psi}_j$ operator the outputs are all transformed similarly. These properties of the learned operators confirm that the model is working correctly – they are essentially the prior constraints that have been used in the estimation of optical flow since its early days, proposed by [Horn and Schunk, 1981] to overcome the under-constrained nature of the flow problem. Since many motions are caused by light reflecting from rigid surfaces, flow fields are usually smooth. This fact implies that intensity measurements can be pooled over some region to obtain two displacement estimates for every intensity measurement. However, some important questions remain, which, left unanswered, make it difficult to feel confident in estimates obtained this way:

- How large a region should be pooled over?
- How do we avoid pooling over occlusion edges, where the flow should certainly not be smooth?

In this case of the model proposed here, these questions can be answered naturally. The smoothness property has emerged by maximizing the log probability of a model that could

have learned something dramatically different, and for this reason, as we increase J, the model will attempt to describe more nuances of the data. In particular, we expect it to develop a mechanism for dealing with occlusion boundaries.

The learned operators are not obviously dominated by first-order oriented spatial derivatives that simply shift the input in a particular direction by less than a pixel. Since locally, most image transformations look like translation, one might assume that first-order differential translation operators would be a prominent mode that the model would be drawn towards. However, the characteristic form for an oriented first-order spatial derivative is evident in the weights for each output pixel in only one of the nine functions. In every other case, there are multiple oscillations in these weights: they do not simply change from bright to dark along any line crossing the patch. This effect grows in strength as the number of basis elements is increased. Figure 2.3 shows functions learned from a model with $J = 16$, and, remarkably, none of the learned functions look like first-order spatial derivatives. In this larger model, one function learned to localize the effect of its transformation to the upper-left corner of its output, suggesting that although spatial consistency is dominant, there are other prevalent patterns waiting to be learned as J is increased further. Indeed, strong differences in the effect of a given $\mathbf{\Psi}_j$ across the patch have been observed for $J = 49$. Often, a sharp horizontal, vertical, or diagonal boundary neatly partitions the effect of the transformation to only one half of its input.

It is evident that the learned functions have tiled the space of translation operators by learning 1st, 2nd, 3rd, and possibly higher-order spatial derivatives. These are combined together additively to effect precise sub-pixel shifts, and to well-approximate shifts more distant than one pixel. Recall that in Equation 2.6, we truncated our expansion of the matrix exponential to two terms. Were only first-order spatial derivatives learned in our $\mathbf{\Psi}_j$ matrices, a three term truncation would include terms of the form $\frac{1}{2}\mathbf{\Psi}_j\,\mathbf{\Psi}_j$ in the sum, implicitly producing second-order spatial derivatives. Since these higher order terms have been truncated, the model has learned their structure in the lower-order terms. Obviously, this comes at a price: instead of learning independent actions in each basis element, the basis elements must now have correlated activation. Through the emergence of this effect, the data telling us that a more compact representation could be achieved by not truncating, or performing a non-linear transformation of the input to a space where a linear operator better approximates shift.

Due to the high degree of spatial locality learned in the $\mathbf{\Psi}_j$ functions, the weights themselves seem to follow a sparse distribution, preferring to be at zero, with heavy tails. We incorporate this characteristic into the model itself, using a Laplacian prior on the values of $\mathbf{\Psi}_j$. This prior is simply added to the energy function, where the $\mathrm{vec}(\cdot)$ operator simply stacks the columns of its matrix argument on top of one another, yielding a vector:

$$E = \sum_t \frac{1}{2\sigma^2}||\mathbf{x}(t+1) - (\mathbf{I} + \sum_j \mathbf{\Psi}_j\, c_j(t))\,\mathbf{x}(t)||_2^2 + \gamma\,||\mathbf{c}(t)||_1 + \alpha\,||\,\mathrm{vec}(\mathbf{\Psi}_j)||_1\,. \qquad (2.11)$$

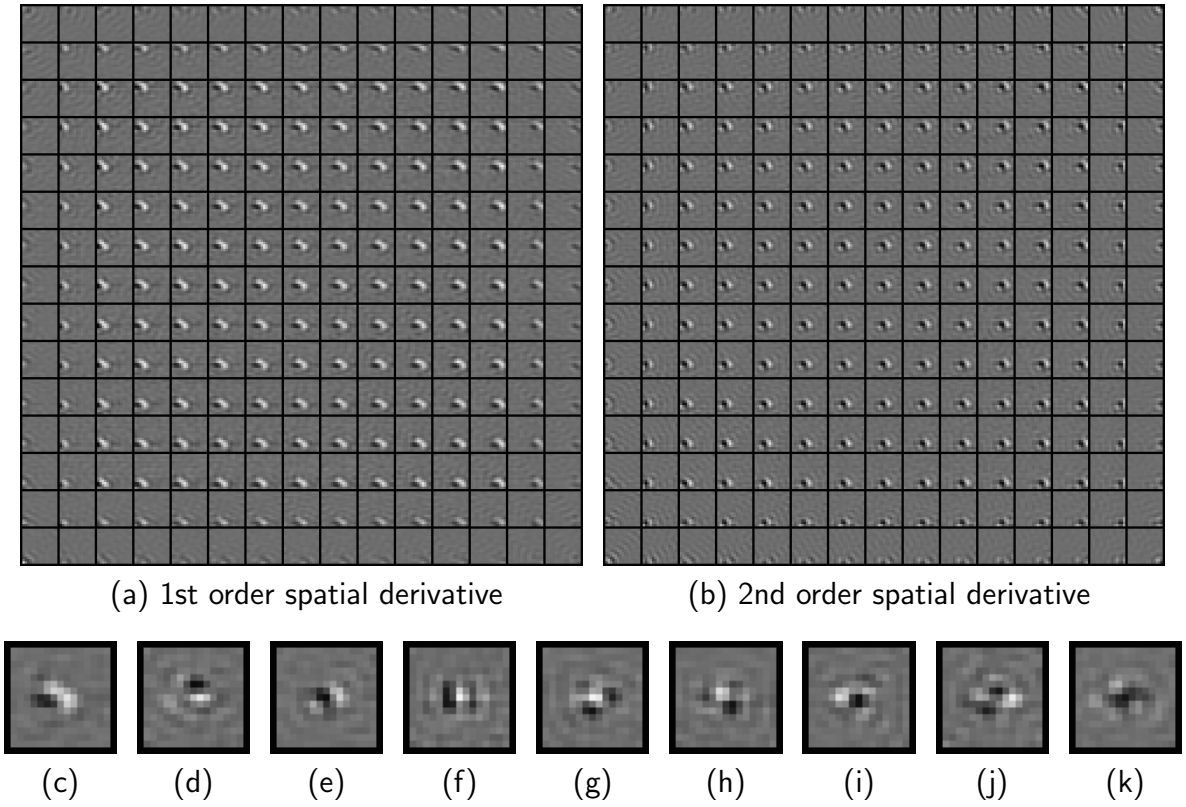


Figure 2.1: **Learned pixel space transformations for natural movies.** (a) and (b) each display one of the nine $\mathbf{\Psi}_j$ transformation basis elements learned in a model with $P = 15^2$, $L = 100$, $J = 9$ and a border of one pixel. Each basis function is displayed in a 2d x 2d embedded space where the connection weights from frame t to an output pixel in frame $t + 1$ are arranged in a square centered at the output pixel position. The function displayed in (a) has learned what appears to be a first-order spatial derivative that is repeated uniformly across the entire input space. That is, there is a spatial symmetry to the way each output pixel is derived from the input. This symmetry is evident in all of the basis functions learned in this model. The function displayed in (b) has learned what appears to be a second-order spatial derivative. In (c-k), the center output pixel has been extracted from each of the nine learned basis elements, and manually arranged (by eye) in order of increasing spatial derivative order. The function displayed in (c) seems to be the only first-order derivative, followed by three second-order derivatives in (d-f), and then higher-orders in (g-k).

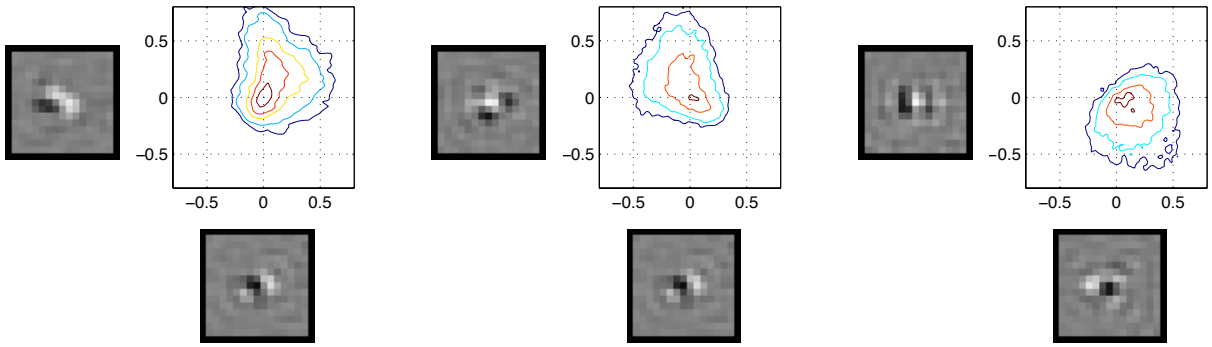


Figure 2.2: **Joint dependencies between transformation basis elements.** In a subset of pairs of functions taken from the model displayed in Figure 2.1, joint histograms of MAP estimates and their contours of equal log-probability are plotted. The prior (not shown) has diamond shaped equal log-probability contours, with the points of the diamond aligned to the axes. Since we are plotting MAP-estimates here, if the prior were a perfect fit to the data, we would expect the contours to be diamond-shaped, but somewhat bowed in towards zero. However, many of the learned transformation basis elements have marginal MAP-estimated coefficient distributions that are primarily one-sided; that is, their coefficients strongly prefer to be positive or negative. Also, there are still significant dependencies between the coefficients in the learned representation. In (a), we see such dependency between a first-order spatial derivative and a second-order spatial derivative with a similar orientation: there is significantly more mass in the second quadrant than would be expected under an independence assumption. In (b), we see a similar effect across a second-order spatial derivative and a higher-order spatial derivative. In (c), we see it between two higher-order derivatives. Such dependencies are consistent with the hypothesis that the model is compensating for the first order Taylor series approximation of the matrix exponential by learning basis elements for the higher-order terms.

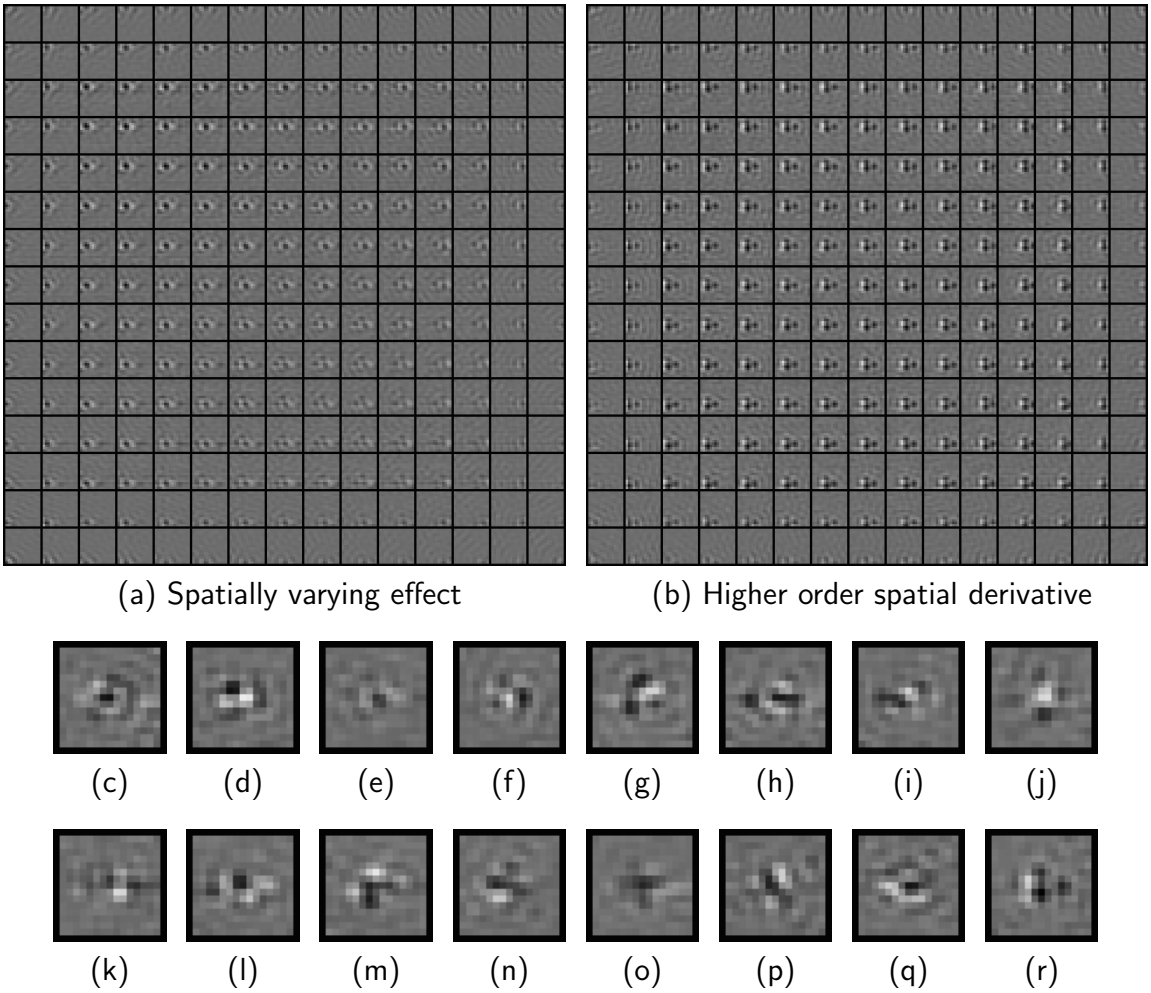


Figure 2.3: **Learned pixel space transformations for natural movies.** (a) and (b) each display one of the 16 $\mathbf{\Psi}_j$ transformation basis elements learned in a model with $P = 15^2$, $L = 100$, $J = 16$ and a border of one pixel. The function displayed in (a) has learned a spatially varying pattern that is repeated, but its effect decays in the output pixels towards the lower right. This type of break in spatial symmetry was not evident in the smaller model with $J = 9$. The function displayed in (b) has learned a higher-order spatial derivative that appears to higher order than any of the functions learned in the model with $J = 9$. In (c-r), the center output pixel has been extracted from each of the 16 learned basis elements, and manually arranged (by eye) in order of increasing spatial derivative order.

The scalar α is set to 0.001, and the previously described learning procedure is used to adjust randomly initialized $\mathbf{\Psi}_j$. With this prior, the model converges much more quickly, enabling experimentation with much larger models. In Figure 2.4 we show four of the learned $\mathbf{\Psi}_j$ for $P = 900$, $L = 900$, $J = 49$ and an extremely conservative masking boundary of 10 pixels on each edge. That is, each $\mathbf{\Psi}_k$ maps a 30×30 patch in frame t to a 10×10 patch in frame $t + 1$. Two particularly interesting aspects of the solution are: (1) the first and third basis functions perform horizontal translations of greater than one pixel; motions of this magnitude cause motion blur in the movie, which is evident in the elongation of the filters along the direction of motion, and (2) the fourth basis function has become localized to operate only in the upper and lower right corners of the image; other basis functions (not shown) became localized to other regions.

2.3 Factoring image sequences

If we are to separate the time-invariant components of a movie (i.e., the objects) from their time-varying positions, our model naturally requires two components: an invariant part that denotes the features of the image, and a variant part that captures changes in the image due to object motion or observer self-motion. In the previous section, from the solution to a linear ODE we obtained a model of natural image sequences that captures local perturbations about a point in time using a linear transformation operator. Next, we show how a two component model can be obtained using these operators. By substituting Equation 2.5 into Equation 2.7, we arrive at the following forward model:

$$\mathbf{x}(t + \delta) = (\mathbf{I} + \delta \sum_j \mathbf{\Psi}_j \, c_j(t)) \, \mathbf{x}(t) + \mathbf{n}(t) \, . \tag{2.12}$$

In this model, the observed variables are $\mathbf{x}(t + \delta)$ and $\mathbf{x}(t)$. There are two sets of latent variables: $\mathbf{c}(t)$, which encodes the time-varying transformation, and $\mathbf{n}(t)$, that accounts for parts of the image sequence that cannot be explained by the transformation model. In some sense, this is a two component model – one component is $\mathbf{x}(t)$ and the other is the transformation given by $\mathbf{I} + \delta \sum_j \mathbf{\Psi}_j \, c_j(t)$ – but one that does not explicitly achieve a stable, invariant representation. The set of parameters $\mathbf{\Psi}_j$ enable us to compactly describe small perturbations around $\mathbf{x}(t)$, but we cannot easily use this information to bring the entire sequence into register. What we are lacking is a smooth curve $\mathbf{z}(t)$ that, at each time t, gives a value close enough to $\mathbf{x}(t)$ that the model is still able to approximately produce it by transforming $\mathbf{z}(t)$. In this case, $\mathbf{z}(t)$ is a second set of latent variables that must be estimated simultaneously with $\mathbf{c}(t)$. The following model makes this idea explicit:

$$\mathbf{x}(t) = (\mathbf{I} + \sum_j \mathbf{\Psi}_j \, c_j(t)) \, \mathbf{z}(t) + \mathbf{n}(t) \, . \tag{2.13}$$

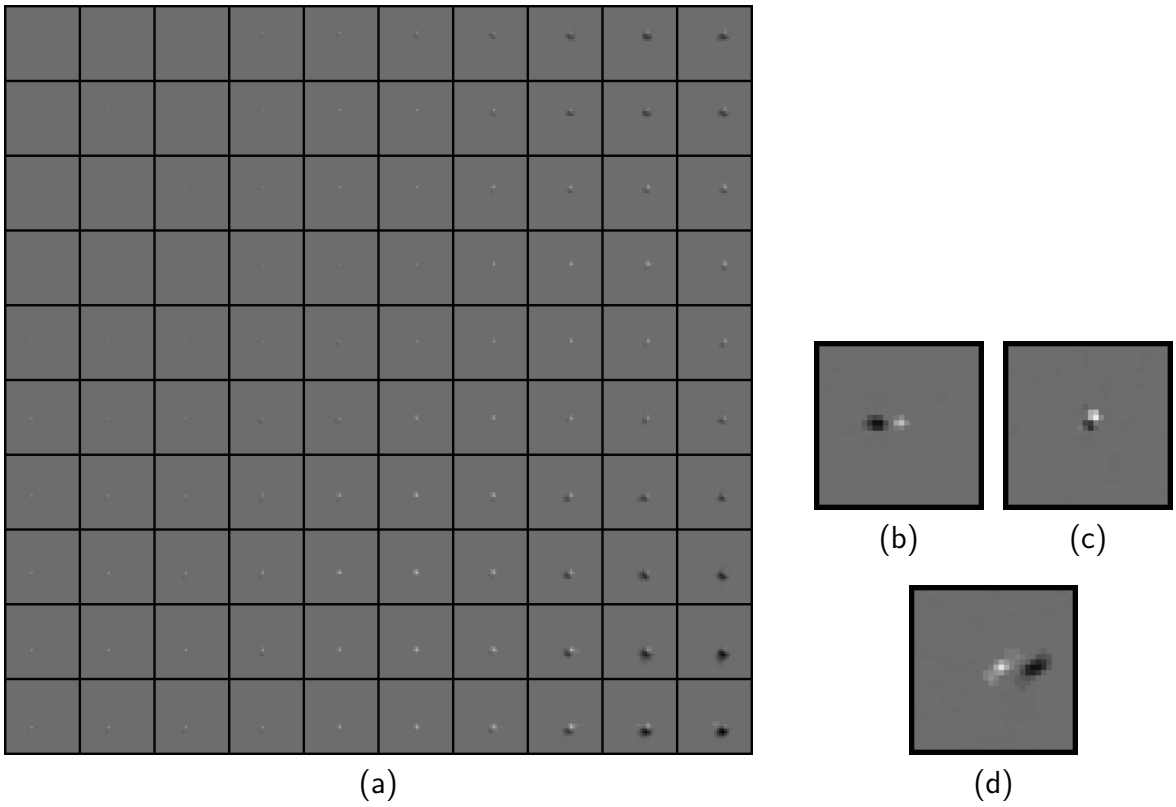


Figure 2.4: **Four of 49 remapping bases, $\mathbf{\Psi}_j$, learned from the transformations contained in natural image sequences.** Most of the 49 basis elements learn to become shifts, tiling the space of distances. Since there are so many basis elements, the model can allocate them to specialized tasks such as shifting the entire patch by a fixed number of pixels. (a) A spatially-varying transformation basis function. Each box within the grid depicts how a pixel in frame $t+1$ is connected to the 30x30 patch from frame t (arranged according to their position in the frame). An i.i.d. Laplace 'weight decay' prior was imposed on the elements of $\mathbf{\Psi}_j$ itself, causing the parts of the basis elements in the error masked region to become completely zeroed out – for that reason we display only the central 10x10 region of the output. (b-d) Because each basis element outputs an even-sized, 10x10 region, there are four center output pixels. We have extracted and enlarged the upper-right output pixel of these central four. (b) A basis element that shifts to the right when its coefficient is negative. A negative coefficient will invert the bright region, weighting it negatively. However, these inverted bright regions will cancel out the contribution of the identity matrix, since the operator is formed via $\mathbf{A}(t) = \mathbf{I} + \sum_j \mathbf{\Psi}_j \, c_j(t)$. The remaining positive weights are all to the left of each output pixel, effecting a shift to the right. (c) A first-order spatial derivative from another spatially uniform basis function (the full function is not pictured). (d) An output pixel from another basis function that shifts to the left by more than a pixel.

To re-iterate an important conceptual change in this new model, note that both $\mathbf{c}(t)$ and $\mathbf{z}(t)$ are latent variables. Thus, each frame of the image sequence is expressed in terms of the product of two latent variables. As discussed in Section 1.4, with this information alone, there is no unique solution for $\mathbf{c}(t)$ and $\mathbf{z}(t)$. To resolve this, we introduce several types of constraints. Some of these constraints are quite straight-forward, such as the parameterization of the function $\mathbf{A}(t) = (\mathbf{I} + \sum_j \mathbf{\Psi}_j \, c_j(t))$. Specifying $J < L^2$ reduces the number of degrees of freedom in $\mathbf{A}(t)$, as well as the number latent variables we must estimate in order to fit the model to an image sequence. Furthermore, we plan to learn these parameters from the data, in a probabilistic framework, using a sparse prior on the $\mathbf{c}(t)$ variables. This sparse prior, and our goal of achieving a stable representation are somewhat 'soft' constraints, and are encoded probabilistically. One way this can be done is by adding a term to the energy function: the goal '$\mathbf{z}(t)$ should vary smoothly in time' can be encoded probabilistically by adding a term like $\sum_t ||\dot{\mathbf{z}}(t)||_2^2$. The notion of using smooth regularizers for under-constrained optimization problems is by no means a new idea. However, it is worth pointing out that we are not blindly regularizing, but using stability to discover invariants – Foldiak, Hyvarinen, and others have demonstrated that the principle of stability can be used as a 'grouping' cue, similar to sparsity [Foldiak, 1991; Wallis and Rolls, 1997; Wiskott and Sejnowski, 2002; Einhauser *et al.*, 2002; Hyvarinen *et al.*, 2003]. Explicit grouping cues have also been used before in this way to learn functions that can transform groups of related objects to an underlying invariant [Miller *et al.*, 2000]. By utilizing a linear decomposition for the $\mathbf{z}(t)$ component of our model,

$$\mathbf{z}(t) = \mathbf{\Phi} \, \mathbf{a}(t) \,, \tag{2.14}$$

and imposing a sparse prior on these new latent $\mathbf{a}$ variables, the learned parameters in $\mathbf{\Phi}$ will function as another set of regularizing constraints. Underlying our approach with both of these parameterized representations is the assumption that images, their underlying invariants, and image transformations have a sparse structure that occupies only a small fraction of the possible parameter space. Furthermore, we assume that relevant directions in this space can be captured in a dictionary of frequently occurring elements that correspond to meaningful causes in the environment. Substituting the decomposition given by Equation 2.14 into Equation 2.13 yields the following factorization model:

$$\mathbf{x}(t) = (\mathbf{I} + \sum_j \mathbf{\Psi}_j \, c_j(t)) \, \mathbf{\Phi} \, \mathbf{a}(t) + \mathbf{n}(t) \,. \tag{2.15}$$

To recap, this factorization emerged from the idea of substituting a smooth latent variable for the starting point $\mathbf{x}(t)$ from Equation 2.12. This smooth latent variable, while representing something slightly different than the content of the image sequence at time t, should take on a value close enough to it that the transformation operator we derived from the approximated solution to the original ODE is still rich enough to describe it. Observe that there is another, somewhat different path we can take, which is likely advantageous.

Rather than describing the time evolution of $\mathbf{x}$ directly, we can instead describe the time evolution of a more symbolic, latent representation $\mathbf{d}$, obtained from the decomposition $\mathbf{x}(t) = \mathbf{\Phi}\,\mathbf{d}(t) + \mathbf{n}(t)$. The resulting factorization model is quite similar to Equation 2.15:

$$\mathbf{x}(t) = \mathbf{\Phi}\,(\mathbf{I} + \sum_j \mathbf{\Psi}_j\, c_j(t))\,\mathbf{d}(t) + \mathbf{n}(t)\,, \tag{2.16}$$

with the only difference being in where $\mathbf{\Phi}$ is multiplied. The advantage of this approach is that the transformations produced by $\mathbf{I} + \sum_j \mathbf{\Psi}_j\, c_j(t)$ now operate in a sparse feature space given by the latent $\mathbf{d}(t)$. With a localized, oriented, bandpass basis for $\mathbf{\Phi}$, the activation of a single $\mathbf{d}$ variable will indicate the presence of an edge; therefore, transformations among $\mathbf{d}$ variables will describe coordinated movements of edges. Although the model can be stated simply, learning the parameters $\mathbf{\Phi}$ and $\mathbf{\Psi}_j$ in this two-level hierarchy is a difficult challenge. Thus, in the following sections, we approach our final goal of learning the parameters of this model by proceeding stage-wise, investigating the sequence of increasingly complex models in the order they have been presented in this section.

2.3.1 Learning to transform underlying invariants

From the image basis $\mathbf{\Phi}$ learned in Section 1.6, and the transformation basis $\mathbf{\Psi}_j$ learned in Section 2.2, we can assemble the parameters required by the model described in Equation 2.15. Performing inference in this model factors an image sequence into a stabilized, parsimonious, and hopefully more symbolically meaningful representation though an explaining away procedure. To obtain the stability property, we introduce priors on the time rate of change of the $\mathbf{a}$ and $\mathbf{c}$ variables through two new terms in the energy function:

$$\begin{aligned} E = \sum_t \frac{1}{2\sigma^2}\,||\mathbf{x}(t) - (\mathbf{I} + \sum_j \mathbf{\Psi}_j\, c_j(t))\,\mathbf{\Phi}\,\mathbf{a}(t)||_2^2 + & \qquad \text{(reconstruction)} \\ \lambda\,||\mathbf{a}(t)||_1 + \lambda\,||\mathbf{c}(t)||_1 + & \qquad \text{(parsimony)} \\ \gamma\,||\dot{\mathbf{a}}(t)||_2^2 + \alpha\,||\dot{\mathbf{c}}(t)||_2^2\,. & \qquad \text{(stability)} \end{aligned} \tag{2.17}$$

These two stability properties have slightly different interpretations. Imposing stability on the invariant underlying image $\mathbf{a}$ can be interpreted as a prior belief that objects in the world persist across time – they do not spontaneously appear and disappear. Imposing stability on the transformation variables $\mathbf{c}$ can be viewed as an inertial prior that captures Newton's first law. Note that these stability priors are non-causal, which means that the graphical model representing this energy function now contains both directed and undirected edges. While there is nothing technically wrong with specifying a probability distribution in this way, it does have a more philosophical downside, which is that the latent variables recovered in the factorization are not causal in time.

We learn $\mathbf{\Psi}_j$ under this model, using MAP-EM with $\lambda = 0.1$, $\gamma = 100$, $\alpha = 1$, and the length constraint enforced after each learning step:

$$||\mathbf{\Psi}_j||_{\mathrm{F}} = L \quad \forall\, j\,. \tag{2.18}$$

The $\mathbf{\Psi}_j$ that result from this training procedure have a coarse qualitative similarity to those learned in the model described in Section 2.2, but here we have demonstrated a functioning training scheme for $\mathbf{\Psi}_j$ that is motivated by recovering an underlying invariant using a prior for stability in time. Figure 2.5 shows the learned $\mathbf{\Psi}_j$ for a model with a large patch size: $P = 19^2$, $L = 100$ and $J = 9$, with a border of two pixels masking errors at the edges of the frame. The solution that emerges is again 1st, 2nd, and 3rd order spatial derivatives that operate symmetrically across the whole patch – there is little localization of the transformation to any specific part of the patch, even in the large $P = 19^2$ case. However, such localization does appear to begin to emerge when $J = 25$. Unfortunately, the number of model parameters in this case is so large that it takes an extremely long time to optimize them. However, they do indicate that spatially localized transformation operators emerge in a factorization model with a linearized transformation, even with no L1 'weight decay' prior. This is an important result, because it means that by tuning the optimization further, this framework will enable us to learn richer motion priors than simply spatial smoothness. Note that for $P = 19^2$, 100 PCA components ($L = 100$) capture 97.97% of the variance in the image sequence, which is still quite significant, though some loses of high spatial frequencies are evident.

After learning $\mathbf{\Psi}_j$, we perform inference on the sequence shown in Figure 2.6, under the model given by Equation 2.17, using the coarse to fine procedure described in Appendix 2.B. Figure 2.7 shows the image sequence and reconstruction under the model, and Figure 2.8 shows the underlying invariant the model extracts from the sequence. Figure 2.9 shows the whitened, dimensionality reduced representation of the image sequence, compared to the stabilized representation under the model.

2.3.2 Learning to transform features of an underlying invariant

Having learned the structure of natural transformations in pixel input vectors, we now turn to the problem of learning the structure of transformations between feature vectors. The model we propose is nearly identical to the previous model, except for the order of multiplication for $\mathbf{\Phi}$:

$$\mathbf{x}(t) = \mathbf{\Phi}\left(\mathbf{I} + \sum_j \mathbf{\Psi}_j\, c_j(t)\right)\mathbf{d}(t) + \mathbf{n}(t)\,. \tag{2.19}$$

Our motivation for learning in feature space is that it is a more natural domain for expressing certain kinds of transformations as a basis function expansion in the linearization of the exponential operator. To illustrate this idea, consider the relationship of our proposed model

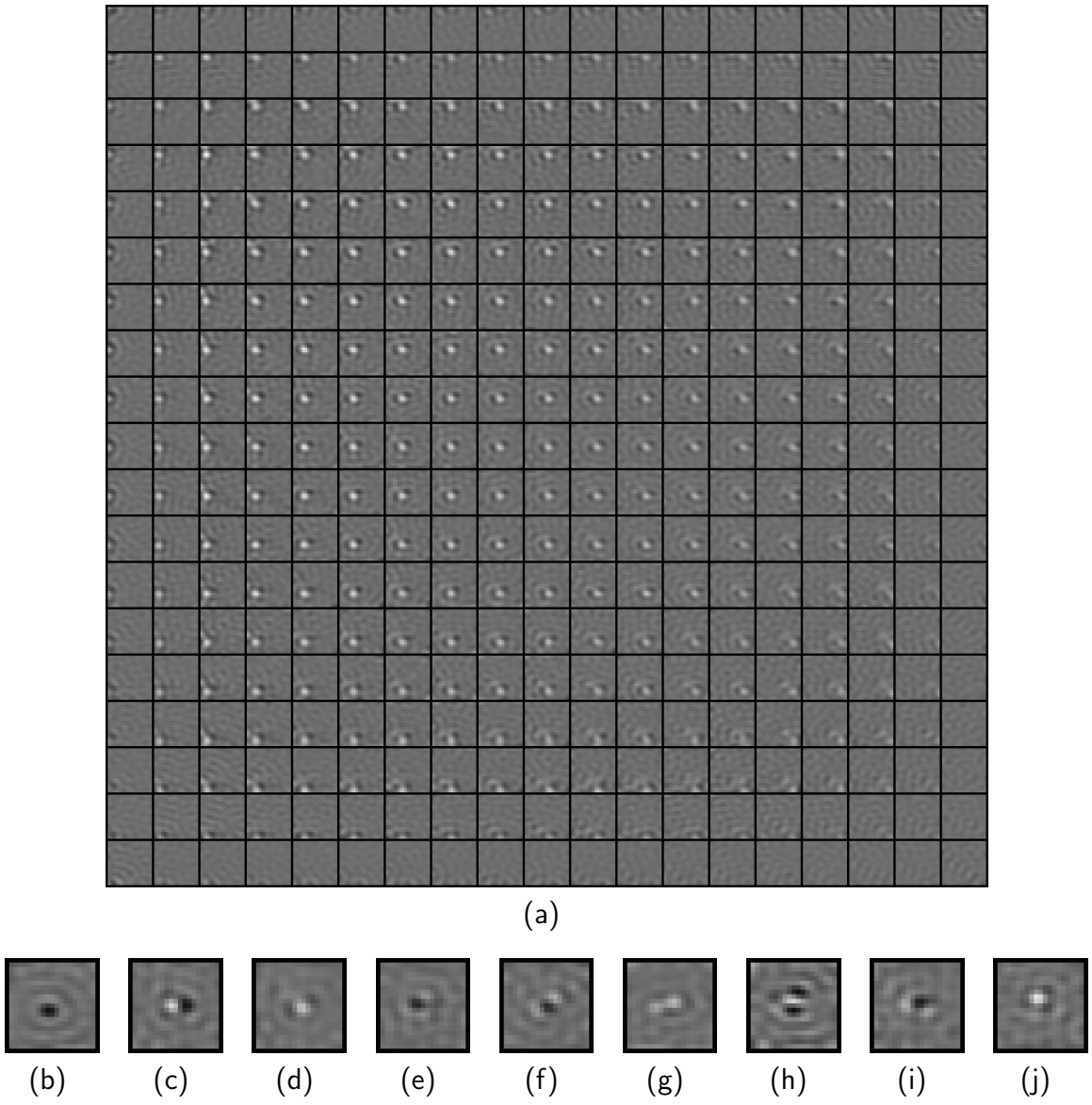


Figure 2.5: **Transformations learned in a factorization model with 19x19 pixel image sequences.** Remarkably, the transformation basis functions learned in the factorization model, which has a latent invariant that must be inferred simultaneously with the transformation, are remarkably similar to those learned in the frame-to-frame transformation model. Nine learned $\mathbf{\Psi}_j$ for $P = 19^2$, $L = 100$, $J = 9$, with a border of two pixels. (a) One of the learned functions has a spatially varying effect which decays toward the lower-right. (b-j) The weights connecting the center output pixel in each of the nine learned functions, manually ordered by increasing spatial derivative order.

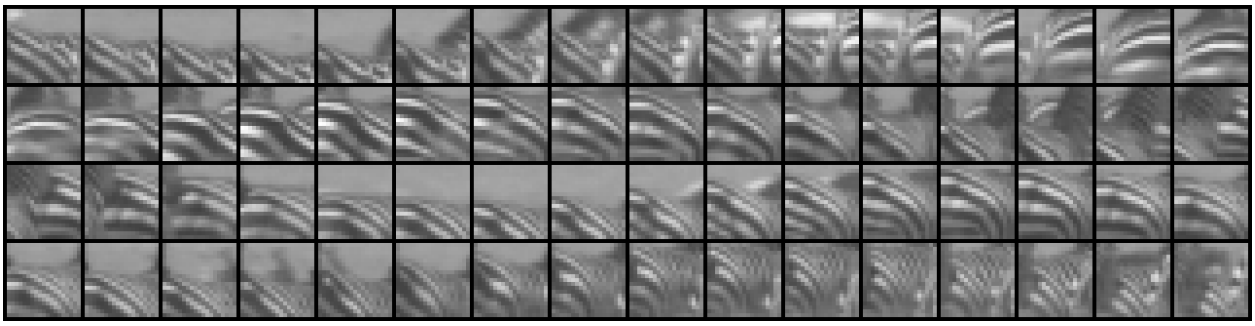


Figure 2.6: **An example of the image sequence data used to train the model.** We perform inference on this sequence under the model by computing MAP estimates the latent $\mathbf{a}(t)$ and $\mathbf{c}(t)$ variables. The scene depicts a running zebra, though only a small portion of it is visible in a small patch. There is clear continuity visible between frames at a symbolic level, but also dramatic changes at the level of pixels.

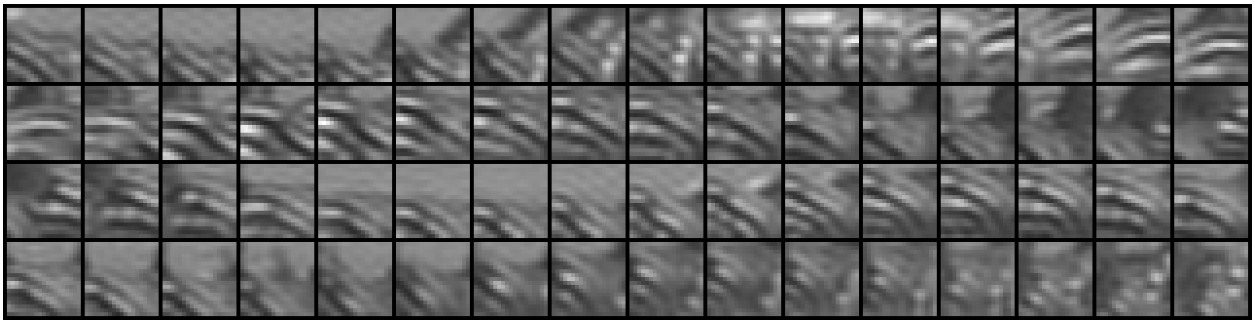
to the bilinear models of [Freeman and Tenenbaum, 1997; Tenenbaum and Freeman, 2000; Grimes and Rao, 2005]. They can be equated to each other by first absorbing our $\mathbf{I}$ term into a new $\mathbf{\Psi}_j$ basis element, with a corresponding c_j coefficient that is always equal to 1. Then, pre-compute

$$\Gamma_{lkj} = \sum_m \phi_{lm}\, \psi_{mkj}\,. \tag{2.20}$$

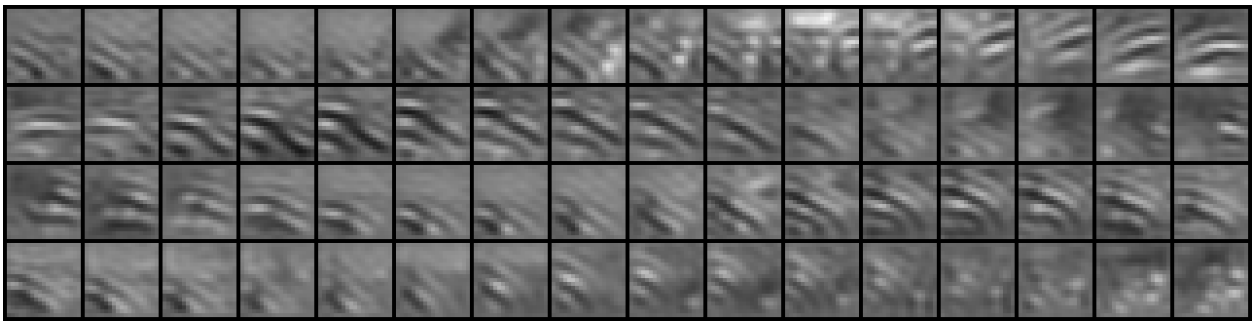
This gives a bilinear model with the same for as a dynamic routing circuit [Olshausen *et al.*, 1993]:

$$\mathbf{x}(t) = \sum_j \mathbf{\Gamma}_j\, c_j(t)\, \mathbf{d}(t)\,. \tag{2.21}$$

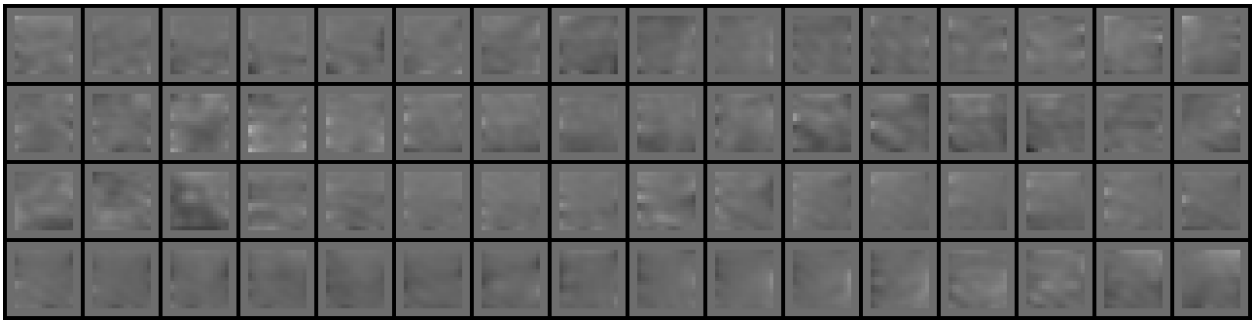
With M image basis vectors and J transformation basis vectors, this model has $L^2J + LM$ free parameters, which can quickly become quite large. However, by not collapsing the sum over m, taken above to equate the two models, we gain an explicit separation of the image and transformation bases. Furthermore, although there are L^2J+LM free parameters in our model and only LKJ in theirs, we expect that the L^2J parameters in our model will be mostly zeros, whereas in the other bilinear models, all of the parameters contain significantly non-zero 'pixel' values. We believe that this fact will enable our learning procedure to converge quickly, despite the large number of free parameters. The feature space transformation model



(a)



(b)



(c)

Figure 2.7: **Example image sequence reconstruction.** (a) The 64 frame example image sequence, as represented by 100 PCA components. (b) The model's reconstruction of the sequence from the MAP estimate of the latent variables. (c) The MAP estimate does not perfectly reconstruct the input; shown here are the reconstruction errors, which are mostly concentrated at the borders, where new content entering the patch is difficult to describe in terms of the invariant. A two pixel border was used to train this model, so there are zero errors along the very edge.

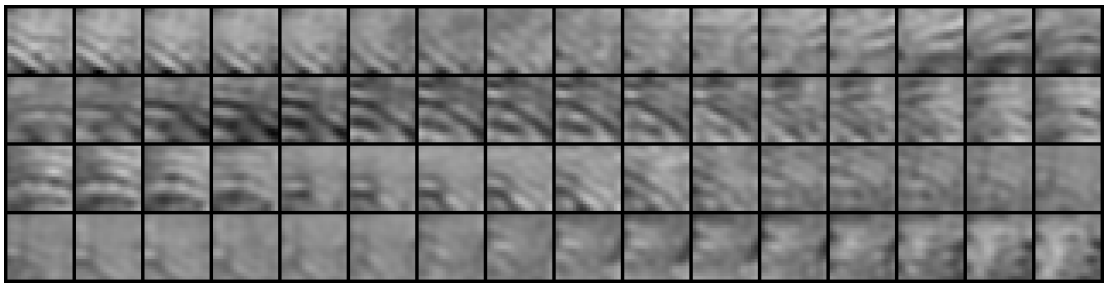


Figure 2.8: **Visualizing the invariant.** The invariant component underlying the sequence of images, that is extracted by performing inference under the model, can be visualized by computing $\mathbf{\Phi}\,\mathbf{a}(t)$ and passing it through the unwhitening matrix. The bilinear model is somewhat limited in its capabilities, and when visualized as an array of frames this way, it is difficult to see what has been stabilized in the image. In the original movie, the pattern of zebra stripes is undergoing considerable translation as the zebra gallops. The translation is too dramatic to completely stabilize, so what the model has done is break the invariant into segments, each of which are stable; at their boundaries, the segments neatly fade in and out, with one stable pattern replacing another. The image sequence $\mathbf{x}(t)$, the invariant component $\mathbf{\Phi}\,\mathbf{a}(t)$, and the time-varying transformation $\mathbf{I} + \sum_j \mathbf{\Psi}_j\, c_j(t)$ can be viewed as movies by clicking on the following links: $\mathbf{\Phi}\,\mathbf{a}(t)$, $\mathbf{x}(t)$, $\mathbf{I} + \sum_j \mathbf{\Psi}_j\, c_j(t)$.

is given by the following energy function:

$$
\begin{aligned}
E = \sum_t \frac{1}{2\sigma^2} \left|\left|\mathbf{x}(t) - \mathbf{\Phi}\left(\mathbf{I} + \sum_j \mathbf{\Psi}_j\, c_j(t)\right)\mathbf{d}(t)\right|\right|_2^2 + \qquad & \text{(reconstruction)} \\
\lambda \left|\left|\mathbf{a}(t)\right|\right|_1 + \lambda \left|\left|\mathbf{c}(t)\right|\right|_1 + \qquad & \text{(parsimony)} \\
\gamma \left|\left|\dot{\mathbf{a}}(t)\right|\right|_2^2 + \alpha \left|\left|\dot{\mathbf{c}}(t)\right|\right|_2^2 + \qquad & \text{(stability)} \qquad (2.22)
\end{aligned}
$$

Inference and learning proceed as previously described with $\mathbf{\Phi}$ set to a previously learned 144 function image basis learned as described in Section 1.6. The effect of each learned transformation operator is explored in Figure 2.10. Where $\boldsymbol{\phi}_m$ denotes column m of $\mathbf{\Phi}$, and $\mathbf{e}_m$ denotes the m-th unit vector, each of the nine larger squares in Figure 2.10 shows $\boldsymbol{\phi}_m(\mathbf{I} + \mathbf{\Psi}_j\, c_j)\,\mathbf{e}_m$ for $m \in 1 \ldots 64$ and $j \in 1 \ldots 9$; c_j is scanned across a range of values that correspond to its empirical distribution. The contrast of each individual $\mathbf{\Phi}$ column in the animation has been adjusted by $\kappa_{mj} = \sqrt{\sum_n |\psi_{nmj}|^2}$ to reflect the amount of control $\mathbf{\Psi}_j$ for a given j has over it. Evidently, each transformation operator has learned to phase shift a subset of the columns of $\mathbf{\Phi}$. Some transformation operators are selective for columns with a particular orientation; others are selective to a high or low spatial frequencies.

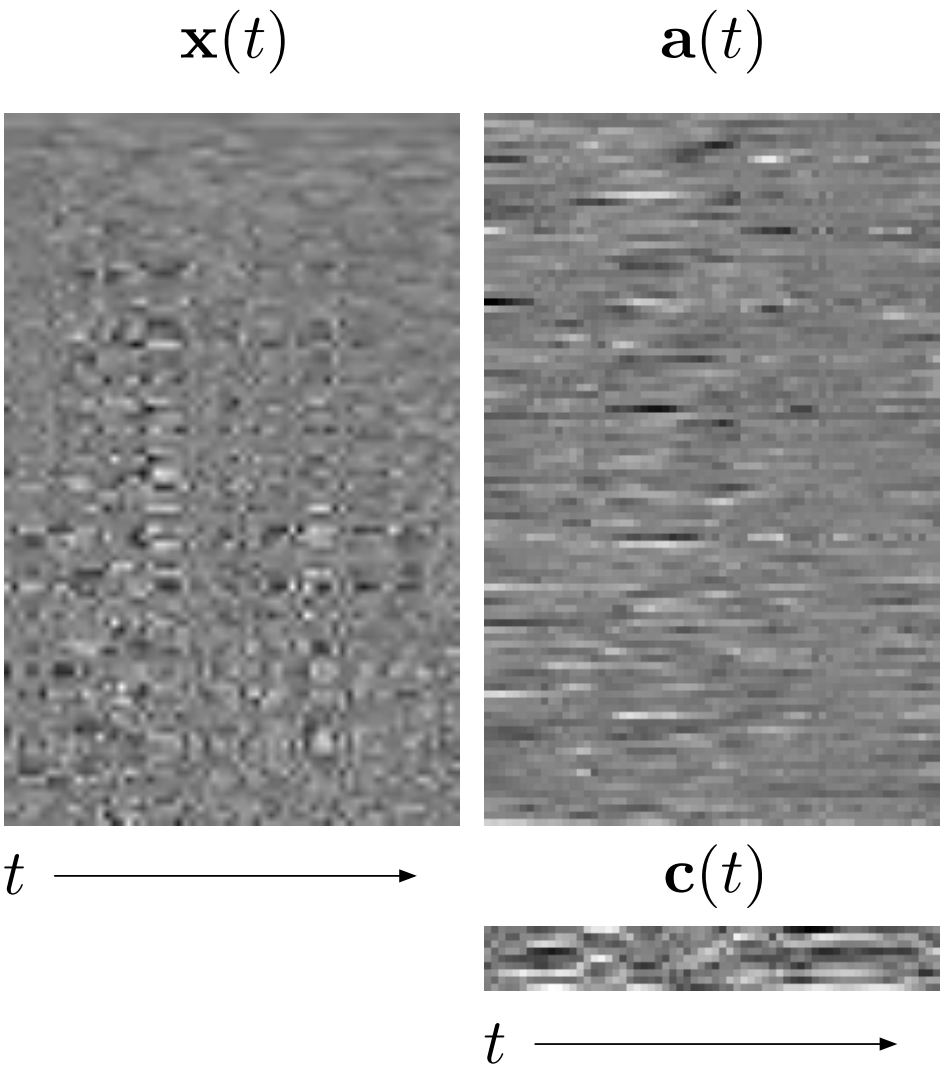


Figure 2.9: **The factored representation of the example image sequence.** (left) Each column shows the 100 PCA component representation of a frame. This is the data that is fed into the model. PCA coefficients often change dramatically between frames, even though at a symbolic level, the visible content does not change much. (middle) Each column depicts MAP estimates for the latent variables in the model representing the underlying invariant. (right) Each column depicts MAP estimates for the latent variables in the model representing how the underlying invariant is transformed into the specific realization of a frame.

This figure must be viewed in a web browser.
Please choose your viewing format by clicking on one of the following links:
gif or avi
(These movies are about 40 MB and will take a moment to load.)

Figure 2.10: **Exploring the effect of the learned Ψ_j operators.** The following movies show the effect of eachj Ψ_j basis function, as c_j is scanned across a range corresponding to its empirical distribution.

2.4 Discussion and concluding remarks

Taking intuition from our world's property of *object constancy*, we have built upon the work of Grimes and Rao, who in turn built on Tenenbaum and Freeman's idea of using bilinear models to separate style and content factors in images. Our starting point for modeling image dynamics was a linear generative sparse coding model, described in Section **??**. A striking deficiency of this model is that the code for an image confounds content and position. In this chaper, we began a treatment of the extraction of motion and form as separate problems, developing a linear generative sparse coding model for transformations, and learning them from movies. In a natural third step, these two linear models were combined into a bilinear model that can also learn without supervision using the established framework. We find in practice that the model is capable of fitting smooth curves to fairly long (64 frame) image sequences, and extracting somewhat meaningful invariance components, such as the zebra stripes shown in Figure 2.8. Thus the model is succeeding in terms of representing an image sequence as a set of features that are stable over reasonable time epochs.

Teasing apart confounded variables is a fundamental difficulty to surmount in vision. This chapter has presented a general principle for approaching such a problem, and demonstrated on a small scale that it works. When applied hierarchically it is likely that the higher order representation will have positive implications for compression and object/pose/movement detection. One shortcoming of the models described that has become evident through our exploration of them is that the first order Taylor expansion is perhaps too coarse an approximation, for the following reason: while the primary mode of variability in the movies is translatory motion and our exponential model can capture such motion perfectly, with a first order Taylor approximation it can only well-approximate short (sub-pixel) translations. In ours and most other natural movies, the temporal sampling frequency is not high enough to characterize all frame to frame changes in terms of sub-pixel translations. Registration experiments with full translation operators give estimates that a significant number of pixels move 4 or more pixels between sampling intervals, though it is impossible to know with certainty since we have no ground truth 3d structure for our movies. Thus, somewhat surprisingly, the assumption of sub-pixel translations between successive movie frames, which

allows a mathematically and computationally convenient linearization, is not practical. In the next chapter, we resolve this by exploring our model of time-varying images in its original form, without approximation.

Appendix 2.A Data pre-processing details

As a pre-processing step, we perform PCA whitening and dimensionality reduction on the individual frames in our image sequences. First, a large batch of data is collected by randomly sampling contiguous sequences of images from the much larger movie. Each frame contains P pixels, and there are T frames in each sequence. The mean (a single scalar) is subtracted from each sequence, and B of these mean-subtracted sequences are collected in a matrix $\mathbf{X} \in \mathbb{R}^{P \times TB}$. The eigen-decomposition of the covariance of $\mathbf{X}$ is computed,

$$\mathbf{U}\,\mathbf{D} = \mathbf{X}\mathbf{X}^{\mathrm{T}}\,, \tag{2.23}$$

where each column of $\mathbf{U} \in \mathbb{R}^{P \times P}$ contains an eigenvector of $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{D}$ is a diagonal matrix of eigenvalues, sorted in decreasing order. Next, a whitening matrix $\mathbf{W} \in \mathbb{R}^{P \times L}$ is computed by dividing the top L eigenvectors by the square root of their corresponding eigenvalue. Denoting these L columns of $\mathbf{U}$ by vectors $\mathbf{u}_{1\ldots L}$,

$$\mathbf{W} = [\frac{\mathbf{u}_1}{\sqrt{D_{11}}} \cdots \frac{\mathbf{u}_L}{\sqrt{D_{LL}}}]\,. \tag{2.24}$$

A corresponding unwhitening matrix $\bar{\mathbf{W}} \in \mathbb{R}^{P \times L}$ is also computed:

$$\bar{\mathbf{W}} = [\mathbf{u}_1\sqrt{D_{11}} \cdots \mathbf{u}_L\sqrt{D_{LL}}]\,. \tag{2.25}$$

Together, these matrices enable a transformation from a P dimensional image patch, to a lower, L dimensional whitened patch, and back again, via projections. That is, if our original image patch is $\tilde{\mathbf{x}}$, the representation we model is

$$\mathbf{x} = \mathbf{W}^{\mathrm{T}}\,\tilde{\mathbf{x}}\,, \tag{2.26}$$

and for visualizing it in the pixel domain, we compute

$$\hat{\mathbf{x}} = \bar{\mathbf{W}}\,\mathbf{x}\,. \tag{2.27}$$

Appendix 2.B Coarse to fine

Consider the frame to frame transformation model described in Section 2.1:

$$\mathbf{x}(t+1) = \mathbf{A}(\mathbf{c})\,\mathbf{x}(t) + \mathbf{n}(t) \tag{2.28}$$

$$\mathbf{A}(\mathbf{c}) = \exp(\sum_j \mathbf{\Psi}_j\, c_j(t))\,, \tag{2.29}$$

with energy function

$$E = \sum_t \frac{1}{2\sigma^2} ||\mathbf{x}(t+1) - \exp(\sum_j \mathbf{\Psi}_j\, c_j(t))\,\mathbf{x}(t)||_2^2 + \gamma\,||\mathbf{c}(t)||_1\,. \tag{2.30}$$

Inference in this model amounts to the computation of

$$\arg\min_{\mathbf{c}(t)} E\,, \tag{2.31}$$

which brings the two frames into registration. Such a registration task requires solving the correspondence problem, and thus we expect it to be particularly challenging to solve robustly for the same reason that applies to optical flow: the generalized aperture problem. Several facts are known about the best way to optimize such objective functions. Specifically, there are two standard techniques used by many algorithms to improve the minimization of optical flow objective functions: graduated non-convexity [Blake and Zisserman, 1987], and coarse-to-fine strategies [Glazer, 1987]. Here we have taken inspiration from these ideas to formulate a mechanism that is highly effective, simple to understand, and computationally tractable in the context of the models we use it to optimize. The main idea comes from an empirical observation that most of the local minima in our energy landscape are due to fast motions or large displacements between the two images being registered. These can be avoided by constructing a sequence of objective functions that are successively better approximations of the true objective function, and solving them one after the other starting each subsequent optimization at the solution to the previous. The sequence of objective functions is constructed by initially limiting the number of PCA components considered in the registration process. The first objective function in the series considers only the component that corresponds to the largest eigenvalue – for natural images, this represents low spatial frequencies. Once the low frequencies have been registered, progressively higher spatial frequencies are considered. All local minima in the energy function can be avoided by starting the optimization using only the lowest spatial frequencies in the input, and this sequence of objective functions can be obtained without any additional computation (recall from Appendix 2.A that they are already ordered this way) – these two factors make it particularly attractive.

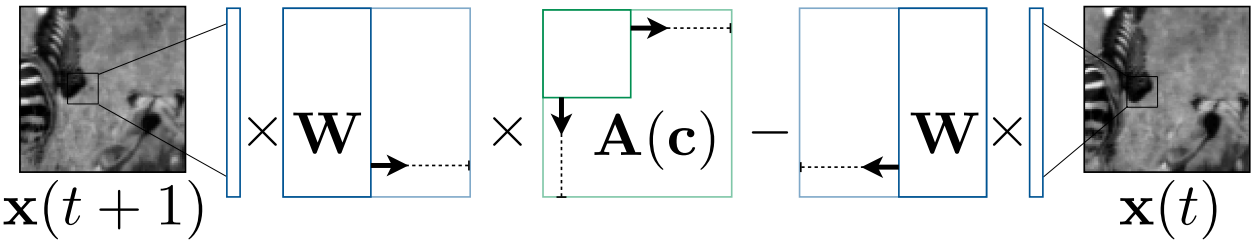


Figure 2.11: **Coarse-to-fine registration procedure.** To avoid poor registrations obtained by converging to a local minima in the energy landscape, pairs of frames are registered in a procedure that progressively builds up power across the frequency spectrum, starting at the low end. Due to the $1/f$ falloff in the power spectrum of natural images and the fact that the columns of our whitening matrix $\mathbf{W}$ have been ordered by decreasing eigenvalue magnitude, when an image is projected on $\mathbf{W}$ the resulting coordinates are ordered and represent progressively increasing spatial frequencies. Progressively more coordinates of $\mathbf{x}(t)$ and $\mathbf{x}(t+1)$ are considered by each objective function in the sequence, by increasing the width of $\mathbf{W}$ as indicated by the arrows growing the blue rectangles. A similar growth occurs in the size of the sub-matrix of $\mathbf{A}(\mathbf{c})$ at each of the L energy functions $E(l)$, indicated by the arrows growing the green rectangles. At the beginning of registration, the $\mathbf{c}$ variables are initialized to zero, and $E(1)$ (with only the lowest spatial frequencies) is minimized. Next, this solution for the $\mathbf{c}$ variables is refined by minimizing $E(2)$, which contains additional high spatial frequencies. This process is repeated, by progressively expanding the number of coordinates in the output of the image's projection on the whitening matrix, until all L of the coordinates are considered in the optimization of $E(L)$.

To make this precise, for $\mathbf{x} \in \mathbb{R}^L$, we construct a sequence of L objective functions, denoted $E(l)$ for $l \in \{1, \ldots, L\}$. Each objective function is minimized by initializing $\mathbf{c}_l(t)$ to $\mathbf{c}_{l-1}(t)$, then solving $\arg\min_{\mathbf{c}_l(t)} E(l)$. The initialization for the first objective, $\mathbf{c}_0(t)$, is the vector of all zeros. Each objective function is constructed by considering only the first l coordinates in $\mathbf{x}(t)$ and $\mathbf{x}(t+1)$, and the $l \times l$ sub-matrix of each $\mathbf{\Psi}_j$ that includes the first l rows and columns. Figure 2.11 depicts this procedure, where progressively more coordinates of $\mathbf{x}(t)$ and $\mathbf{x}(t+1)$ are considered by increasing the width of the whitening matrix $\mathbf{W}$, as indicated by the arrows. As a side-note, because of the $1/f$ fall-off in the power spectrum of natural images, any spectral image decomposition whose components are registered in order of decreasing power will automatically implement this mechanism, which makes it seem likely to emerge in biological systems.

We also use this coarse to fine procedure during learning, by initially truncating the sequence of objective functions at some approximate level $\bar{L} < L$, which is increased every 10,000 learning updates. This means that learning only takes place in the $\bar{L} \times \bar{L}$ sub-matrix of each $\mathbf{\Psi}_j$. During the course of learning, more entries of $\mathbf{\Psi}_j$ are adjusted, and eventually $\bar{L} = L$ and each learning update modifies every entry.

Chapter 3

Learning continuous transformation groups from image sequences

It is well known that natural images occupy a small fraction of the space of all possible images. Moreover, as images change over time in response to observer motion or changes in the environment, they trace out particular trajectories along manifolds in this space. It is reasonable to expect that perceptual systems have evolved ways to efficiently model these manifolds, and thus mathematical models that capture their structure in operators that transport along them may be of use for understanding perceptual systems, as well as for engineering artificial vision systems. In this chapter, we derive methods for learning these transport operators from data.

Rather than simply learning a mapping of individual data points to a low-dimensional space, we seek a compact representation of the entire manifold via the operators that traverse it. We investigate a direct application of the Lie approach to invariance [VanGool *et al.*, 1995], utilizing a matrix exponential generative model for transformed images. This is in contrast to previous methods that rely mainly upon a first order Taylor series approximation of the matrix exponential [Rao and Ruderman, 1999; Miao and Rao, 2007], and bilinear models, in which the transformation variables interact multiplicatively with the input [Tenenbaum and Freeman, 2000; Grimes and Rao, 2002; Olshausen *et al.*, 2007]. It is also distinct from the class of methods that learn embeddings of manifolds from point cloud data [Roweis and Saul, 2000; Weinberger and Saul, 2004; Tenenbaum *et al.*, 2000; Belkin and Niyogi, 2002]. We share the goal of learning a model of the manifold which can then be generalized to new data with [Bengio and Monperrus, 2005; Bengio *et al.*, 2006; Dollar *et al.*, 2007].

Our method is essentially an unsupervised manifold learning algorithm that represents a surface through a compact description of operators that traverse it. These operators are based on matrix exponentials, which are the solution to a system of first-order linear differential equations. The matrix exponents are represented by a basis that is adapted to the statistics of the data so that the infinitesimal generator for a trajectory along the

underlying manifold can be produced by linearly composing a few elements.

In Chapter 2 we introduced the idea of modeling short image sequences using first-order linear ODEs. We proposed to model longer sequences through the use of time-varying matrix exponent with a low-dimensional parameterization. However, in the end we coarsely approximated the matrix exponential by truncating Equation 2.3 to two terms, imposing severe limitations on the length of time over which the resulting approximation error is reasonably small. In this chapter, we undo this approximation and model image sequences with continuous transformation groups. As in Chapter 2, we develop algorithms for fitting the model parameters to natural image sequences, and factoring the sequences into an underlying invariant component and its time-varying transformation.

3.1 Continuous transformation model

Here we show how a particular class of transport operators for moving along manifolds may be learned from data. The model is first applied to synthetic datasets to demonstrate interesting cases where it can recover topology, and that for more difficult cases it neatly approximates the local structure. Subsequently, we apply it to time-varying natural images and show that it learns to a more compact representation of the dynamics of appearance compared to that which is learned by its first order Taylor approximated counterpart.

As in Chapter 2, let us consider an image of the visual world as a point $\mathbf{x} \in \mathbb{R}^L$. We describe the evolution of $\mathbf{x}$ in terms of temporal changes δ from a starting time t by the solution to a linear ODE:

$$\mathbf{x}(t+\delta) = \exp(\mathbf{A}\,\delta)\,\mathbf{x}(t)\,. \tag{3.1}$$

The matrix $\mathbf{A}$ is a linear operator capturing some action in the environment that transforms the image. Such an action belongs to a family that occupies a subspace of $\mathbb{R}^{L\times L}$ given by

$$\mathbf{A} = \sum_j \mathbf{\Psi}_j\, c_j\,, \tag{3.2}$$

with $\mathbf{\Psi}_j \in \mathbb{R}^{L\times L}$. The amount of a particular action from the dictionary $\mathbf{\Psi}_j$ that occurs is controlled by the corresponding c_j. To capture the fact that natural movies contain sequences of many actions which share structural properties, we express the action as a time-varying function of a small set of coefficients, and add a noise term $\mathbf{n}(t)$ to account for structure in natural image sequences that cannot be explained by the model:

$$\mathbf{x}(t+\delta) = \exp(\sum_j \mathbf{\Psi}_j\, c_j(t))\,\mathbf{x}(t) + \mathbf{n}(t)\,. \tag{3.3}$$

Beginning at time t, a vision system takes an image $\mathbf{x}(t)$, and then makes repeated observations at a fixed interval; thus, for some unit of time, our measurements always occur at

integer values of t. Given an initial scene $\mathbf{x}(t)$ and an action in the environment $\mathbf{A}$, the progression of time increases δ and traces out a continuously differentiable manifold of images given by Equation 3.1, which we observe periodically. Our goal is to learn an appropriate set of bases, $\mathbf{\Psi}_j$, that allow for a compact description of observed frame to frame transformations by training on many pairs of related observations.

In terms of modeling capacity, exponential operators are far more expressive than their linear counterparts. Thus, if learned properly, a reasonable expectation would be for them to capture the structure of natural image transformations to a greater degree, but at a far greater computational cost. Although it is still slightly more computationally expensive than its linearized approximation, in the next section we see that the partial derivatives of the energy function for this continuous transformation model have a simple form that may be computed efficiently.

3.1.1 Learning

The model parameters $\mathbf{\Psi}_j$ are learned by maximizing log-likelihood using the MAP-EM algorithm, described in Section 1.6, and briefly reviewed here. Consider two 'close' states of the system in isolation. Let $\mathbf{x}(t)$ be our initial condition, and $\mathbf{x}(t+1)$ be a second observation. These points are related through an exponentiated matrix that itself is composed of a few basis elements, plus zero-mean white i.i.d. Gaussian noise, $\mathbf{n}(t)$:

$$\mathbf{x}(t+1) = \exp(\sum_j \mathbf{\Psi}_j \, c_j(t)) \, \mathbf{x}(t) + \mathbf{n}(t) \tag{3.4}$$

We assume a factorial sparse prior over the transform variables $\mathbf{c}(t)$ of the form $P(c_m(t)) \propto \exp(-\gamma \, |c_m(t)|)$. The variance of the noise is σ^2, and thus the negative log of the posterior probability of the data under the model is given by

$$E = \sum_t \frac{1}{2\sigma^2} ||\mathbf{x}(t+1) - \exp(\sum_j \mathbf{\Psi}_j \, c_j(t)) \, \mathbf{x}(t)||_2^2 + \gamma ||\mathbf{c}(t)||_1 \, . \tag{3.5}$$

Given two data points, the solution of the $\mathbf{c}(t)$ variables which relate them through $\mathbf{\Psi}_j$ is found by a fast minimization of E with respect to $\mathbf{c}(t)$. Learning of the basis $\mathbf{\Psi}_j$ proceeds by gradient descent with respect to E, holding this solution for $\mathbf{c}(t)$ fixed. The $\mathbf{\Psi}_j$ variables are initialized randomly, and adjusted according to $\Delta \mathbf{\Psi}_j \propto -\frac{\partial E}{\partial \mathbf{\Psi}_j}$.

To simplify our notation, let us temporarily consider two sequential points, $\mathbf{x}(0)$ and $\mathbf{x}(1)$ in isolation. The partial derivatives of E w.r.t. $\mathbf{c}$ and $\mathbf{\Psi}_j$ can be cast in a simple form using the spectral decomposition of $\mathbf{A}$, given by $\sum_\alpha \lambda_\alpha \mathbf{u}_\alpha \mathbf{v}_\alpha^{\mathrm{T}}$, with right eigenvectors $\mathbf{u}_\alpha$, left eigenvectors $\mathbf{v}_\alpha$, and eigenvalues λ_α [Ortiz *et al.*, 2001]. Let $\mathbf{U} = [\mathbf{u}_1 \mathbf{u}_2 ... \mathbf{u}_N]$ and

$\mathbf{V} = [\mathbf{v}_1\mathbf{v}_2...\mathbf{v}_N]$. Then

$$\frac{\partial \exp(A)_{ij}}{\partial A_{kl}} = \sum_{\alpha\beta} F_{\alpha\beta} U_{i\alpha} V_{k\alpha} U_{l\beta} V_{j\beta} \,, \tag{3.6}$$

where the matrix $\mathbf{F}$ is given by:

$$F_{\alpha\beta} = \begin{cases} \frac{\exp(\lambda_\beta) - \exp(\lambda_\alpha)}{\lambda_\beta - \lambda_\alpha} & \text{if} \quad \lambda_\beta \neq \lambda_\alpha \\ \exp(\lambda_\alpha) & \text{otherwise.} \end{cases} \tag{3.7}$$

Application of the chain rule yields simplified forms for the partials of E w.r.t. $\mathbf{c}$ and $\mathbf{\Psi}_j$. After computing two intermediate terms $\mathbf{P}$ and $\mathbf{Q}$,

$$\mathbf{P} = \mathbf{U}^{\mathrm{T}} \left(\exp(\mathbf{A})\, \mathbf{x}(0)\, \mathbf{x}(0)^{\mathrm{T}} - \mathbf{x}(1)\, \mathbf{x}(0)^{\mathrm{T}} \right) \mathbf{V} \tag{3.8}$$

$$\mathbf{Q} = \mathbf{V} \left(\mathbf{F} \odot \mathbf{P} \right) \mathbf{U}^{\mathrm{T}} \,, \tag{3.9}$$

the partial derivatives for inference and learning are given by

$$\frac{\partial E}{\partial c_j} = \mathrm{vec}(\mathbf{Q})^{\mathrm{T}}\, \mathrm{vec}(\mathbf{\Psi}_j) + \gamma\ \mathrm{sgn}(c_j) \tag{3.10}$$

and

$$\frac{\partial E}{\partial \mathbf{\Psi}_j} = \mathbf{Q}\, c_j \,, \tag{3.11}$$

where the $\mathrm{vec}(\cdot)$ operator stacks the columns of its matrix argument on top of each other to form a vector. The order of complexity for both derivatives is determined by the computation of $\mathbf{Q}$, which requires an eigen-decomposition and a few matrix multiplications. Note, however, that by simplifying our notation we have glossed over the fact that the composed action operator $\mathbf{A}$ is a function of time, as are $\mathbf{U}, \mathbf{V}, \mathbf{P}, \mathbf{Q}$ and $\mathbf{F}$. Thus, to compute Equations 3.10 and 3.11 over a sequence, these matrices must be computed separately for each pair of sequential points.

3.1.2 Recovery experiments

We first test the model by applying it to simple datasets where the solutions are known: learning the topology of a sphere and a torus, and the local geometry of a Klein bottle, where the topology cannot be captured by first order exponential operators. In these recovery experiments, we add a Gaussian 'weight decay' term to the energy function, which encourages the solution to use only a subset of the operators if possible:

$$E = \sum_t \frac{1}{2\sigma^2} ||\mathbf{x}(t+1) - \exp(\sum_j \mathbf{\Psi}_j\, c_j(t))\, \mathbf{x}(t)||_2^2 + \gamma ||\mathbf{c}(t)||_1 + \zeta \sum_j ||\mathbf{\Psi}_j||_{\mathrm{F}} \,. \tag{3.12}$$

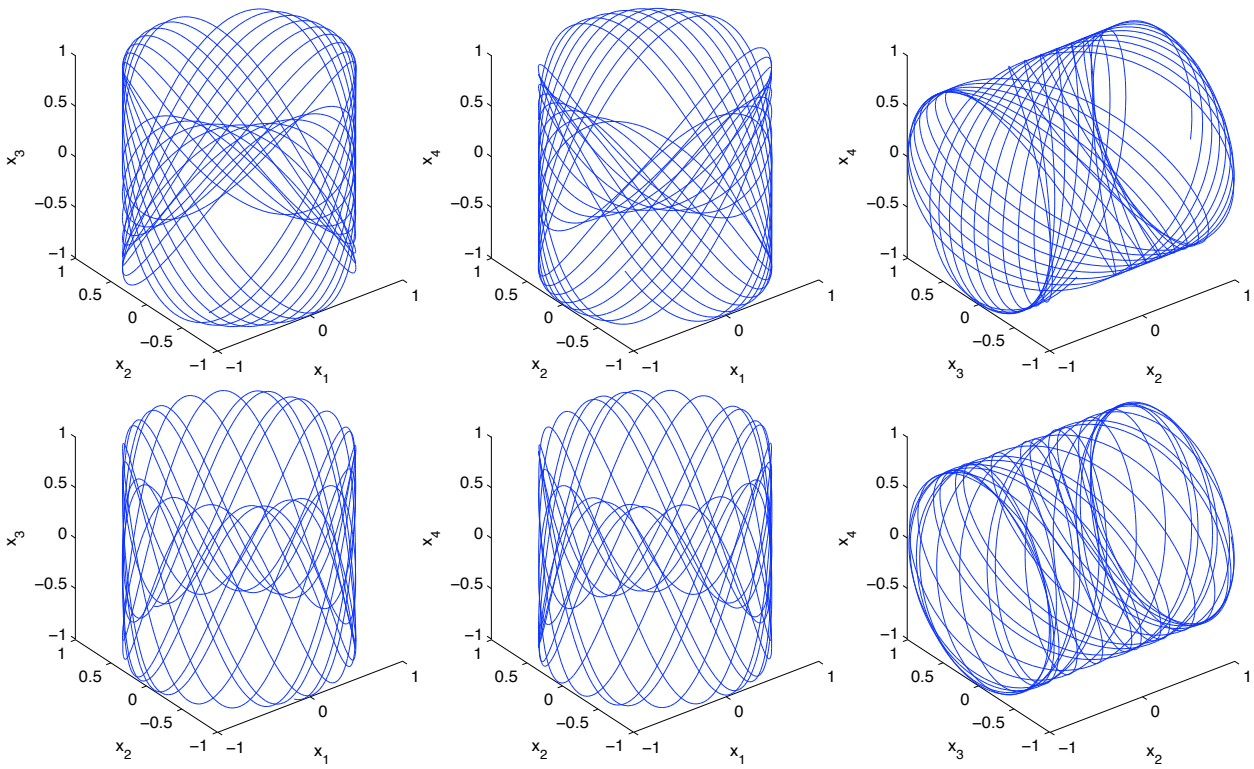


Figure 3.1: **Orbits of learned torus operators.** Each row shows three projections of a $\mathbf{\Psi}_j$ basis element applied to a point on the surface of the torus. The orbits shown are generated by setting $\mathbf{x}(0) = (0, 1, 0, 1)$ then plotting $\mathbf{x}(t) = \exp(\mathbf{\Psi}_j\, t)\, \mathbf{x}(0)$ for $t = [-1000, 1000]$ in projections constructed from each triplet of the four coordinates. In each plot, two coordinates always obey a circular relationship, while the third varies more freely.

Related pairs of points on a torus are generated by first choosing two angles $\theta(0), \phi(0)$ uniformly at random from $[0, 2\pi]$. Two related angles $\theta(1), \phi(1)$ are produced by sampling from von Mises distributions with means $\theta(0)$ and $\phi(0)$, and concentration $\kappa = 5$. For the sphere, we generate the first pair of angles using the normal-deviate method, to avoid concentration of samples near the poles. Though parameterized by two angles, the coordinates of points on these surfaces are 3- and 4-dimensional; pairs of points $\mathbf{x}(t)$ for $t = 0, 1$ on the unit sphere are given by $\mathbf{x}(t) = (\sin\theta(t)\cos\phi(t), \sin\theta(t)\sin\phi(t), \cos\theta(t))$, and points on a torus by $\mathbf{x}(t) = (\cos\theta(t), \sin\theta(t), \cos\phi(t), \sin\phi(t))$.

For the sphere, $L = 3$, thus setting $J = 9$ gives the model the freedom to generate the full space of $\mathbf{A}$ operators. The $\mathbf{\Psi}_j$ are initialized to mean-zero white Gaussian noise with variance 0.01, and $10,000$ learning updates are computed by generating a pair of related points, minimizing E w.r.t. $\mathbf{c}$, then updating $\mathbf{\Psi}_j$ according to $\Delta\mathbf{\Psi}_j = -\eta\, \frac{\partial E}{\partial \mathbf{\Psi}_j}$. In all of the

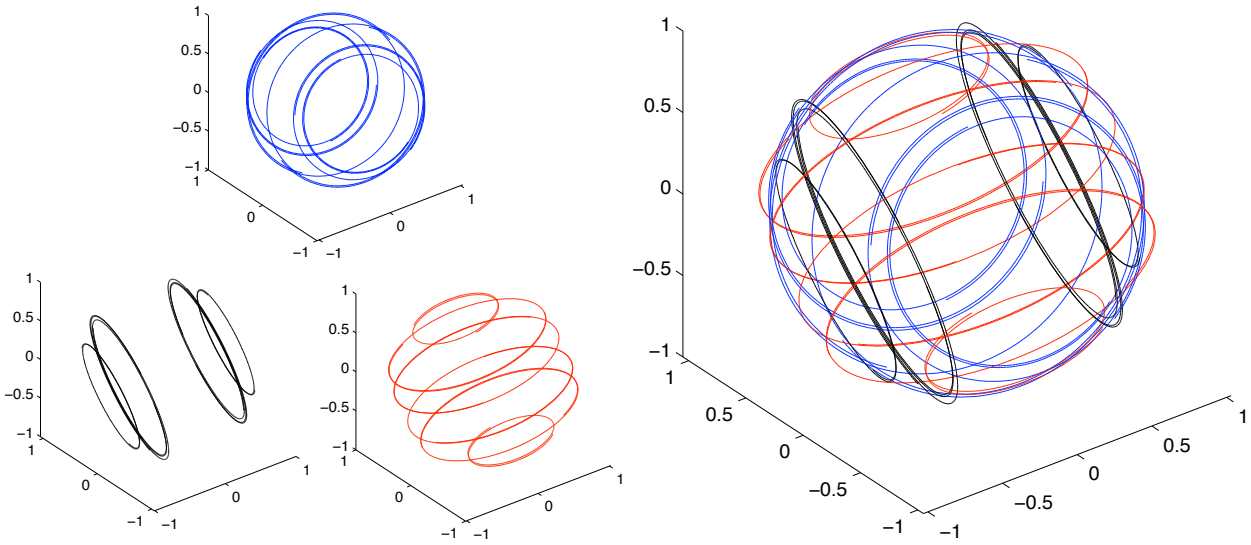


Figure 3.2: **Orbits of learned sphere operators.** (a) Three $\mathbf{\Psi}_m$ basis elements applied to points at the six poles of the sphere, $(1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0)$, and $(0,0,-1)$. The orbits are generated by setting $\mathbf{x}_0$ to a pole, then plotting $\mathbf{x}_t = \exp(\mathbf{\Psi}_m\, t)\, \mathbf{x}_0$ for $t = [-100, 100]$. (b) When superimposed on top of each other, the three sets of orbits clearly define the surface of a sphere.

point set experiments, $\gamma = 0.01$ and $\zeta = 0.0001$. For cases where topology can be recovered, the solution is robust to the settings of γ and ζ – changing either variable by an order of magnitude does not change the solution, though it may increase the number of learning steps required to get to it. In cases where the topology can not be recovered, the influence on the solution of the settings of γ and ζ is more subtle, as their relative values effectively trade-off the importance of data reconstruction and the sparsity of the vector $\mathbf{c}$. We adjust η during learning as follows: when $\Delta\mathbf{\Psi}_j$ causes E to decrease, we multiply η by 1.01; otherwise, we divide by the same amount. When the model has more parameters than it needs to fully capture the topology of the sphere this fact is evident from the solution it learns: six of the dictionary elements $\mathbf{\Psi}_j$ drop out (they have norm less than 10^{-6}). Figure 3.2 shows orbits produced by applying each of the remaining $\mathbf{\Psi}_j$ operators to points on the sphere. Similar experiments are successful for the torus; Figure 3.1 shows trajectories of the operators learned for the torus.

In cases where topology cannot be captured by a matrix exponential operator, we would like the operators to capture local structure. The Klein bottle is a topology we are partic-

ularly interested in modeling, since studies have shown that it is the topology of natural image patches [Carlsson *et al.*, 2007], and is reflected in the structure of cortical orientation maps [Swindale, 1996]. The model gives a locally good approximation to the Klein bottle surface, detailed in Figure 3.4. However, there are many pairs of points on the Klein bottle that cannot be related through an operator whose trajectory exactly tracks the manifold. To see this, recall that the surface of a Klein bottle can be viewed as a 2-dimensional sheet with a periodic horizontal boundary, and a reflected vertical boundary, as depicted in Figure 3.3. Two points related by non-zero horizontal and vertical displacements are connected through a diagonal line on this sheet. Locations A and B on the sheet correspond to the same point, as do C and D, due to the side identifications. The line intersects itself at point E, and at the intersection, the directions of travel are perpendicular. Since the effect of the Lie group operators is entirely determined by $\mathbf{x}$, there can be no operator $\mathbf{A}$ that exhibits this global behavior. However, when given two nearby points, our inference procedure should deliver an $\mathbf{A}$ that can be used to locally interpolate between points along the surface; as the separation distance increases, the interpolation path will begin to slip off. For the case of the Klein bottle, we investigate the extent to which this slippage occurs by attempting to interpolate over sufficiently long segments.

Related pairs of points on a Klein bottle are generated by choosing two angles θ_0, ϕ_0 uniformly at random from $[0, 2\pi]$; two related angles θ_1, ϕ_1 are produced by sampling from two von Mises distributions with means θ_0 and ϕ_0, and concentration $\kappa = 5$. We use the 4-dimensional embedding of [Tompkins, 1941]:

$$\mathbf{x}(t) = (\cos\theta(t)\cos\phi(t),\ \sin\theta(t)\cos\phi(t),\ 2\cos\frac{\theta(t)}{2}\sin\phi(t),\ 2\sin\frac{\theta(t)}{2}\sin\phi(t))\,. \tag{3.13}$$

Our interpolation along the path shown in Figure 3.4 is three times better than linear interpolation (as measured by MSE), and when both techniques have twice as many segments, ours is superior by an order of magnitude. This illustrates the robustness of our algorithm in a case where the underlying data cannot be captured by a linear ODE. A piecewise model, where each piece is approximated by a linear ODE using an $\mathbf{A}$ generated from a learned basis $\mathbf{\Psi}$, gives a sensible answer.

Near-perfect translation operators are recovered from paired synthetic noise images related through sub-pixel 2d translations. Figure 3.5(b-e) shows operators learned with $L = 11^2$, $J = 4$ from data generated in the following way. First, a 15x15 pixel frame $\mathbf{x}(0)$ was drawn from a normal i.i.d. Gaussian distribution, and horizontal and vertical shift parameters p and q were drawn uniformly from $[-1, 1]$. These parameters specify a translation which is applied to the frame to generate a paired frame $\mathbf{x}(1)$ through the following equation:

$$\mathbf{x}(1) = \exp(\mathbf{\Psi}_1\, p + \mathbf{\Psi}_2\, q)\,\mathbf{x}(0)\,, \tag{3.14}$$

where $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ are horizontal and vertical translation operators with wrap-around bound-

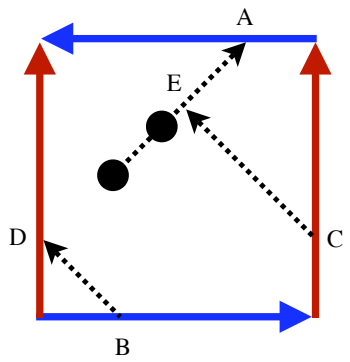


Figure 3.3: **The problem with the Klein bottle surface.** The line connecting the two points intersects itself at E, and at this intersection the line is perpendicular to itself, making clear why a single **A** operator that interpolates between two such points cannot globally stay on the Klein bottle: each operator determines a single direction of movement from an initial condition.

ary conditions; such operators are easy to specify through the use of the Fourier shift theorem. Figure 3.5(a) shows the center row of the horizontal shift operator used to generate the data. After generating paired 15x15 images, the center 11x11 region in each image is extracted to remove the effect of the wrap-around boundary condition which will not occur in natural images.

The independence between vertical and horizontal translation is clearly evident in the learned operators, though no one operator effects exclusively vertical or horizontal translation. Instead, all four operators translate diagonally, with two pairs of operators effecting primarily vertical or horizontal shifts. Compare the second and fourth operators shown in Figure 3.5(c) and (d), which primarily effect horizontal translation, to the structure of the horizontal translation operator used to generate the data, shown in Figure 3.5(a). The learned operators are structurally similar, but do not ring across the entire input patch. Instead, operator 2 rings only to the right of the output pixel, whereas operator 4 rings to the left. A similar asymmetry is evident in operators 1 and 3, the two that primarily translate vertically. These asymmetries occur because the circular boundary condition in the analytical operator was removed training data by clipping out the center 11x11 region. Thus, although only a single operator is required to synthesize horizontal translation with circular boundary conditions, when it is removed two are required: one for each direction.

3.1.3 Natural image sequences

As an intermediate step towards modeling time varying natural images, we investigate the model's ability to learn the response surface for a single complex oriented filter to a moving

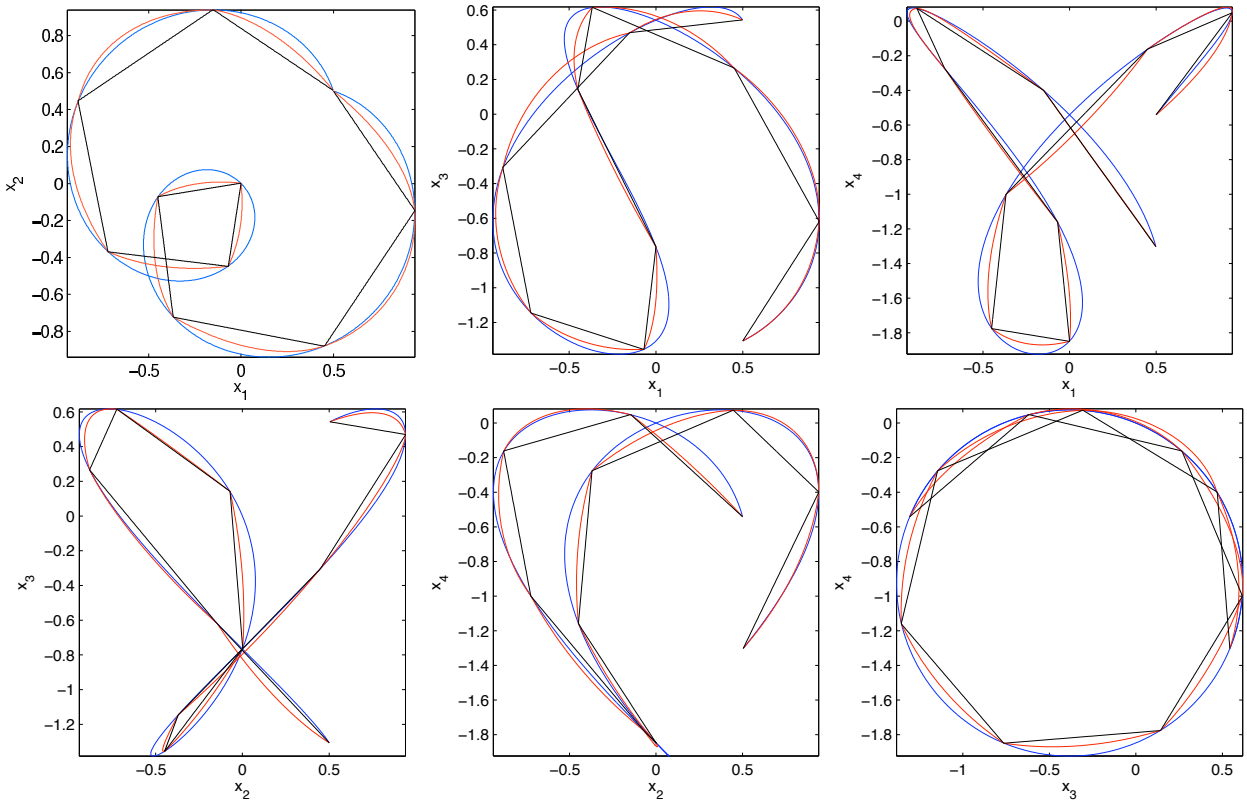


Figure 3.4: **Approximating the surface of a Klein bottle.** Interpolated paths estimated along trajectories anchored to the surface at their endpoints. The red line shows the true surface; the blue is the interpolation given by the model; the black is linear interpolation. Our model does three times better than linear interpolation (as measured by MSE).

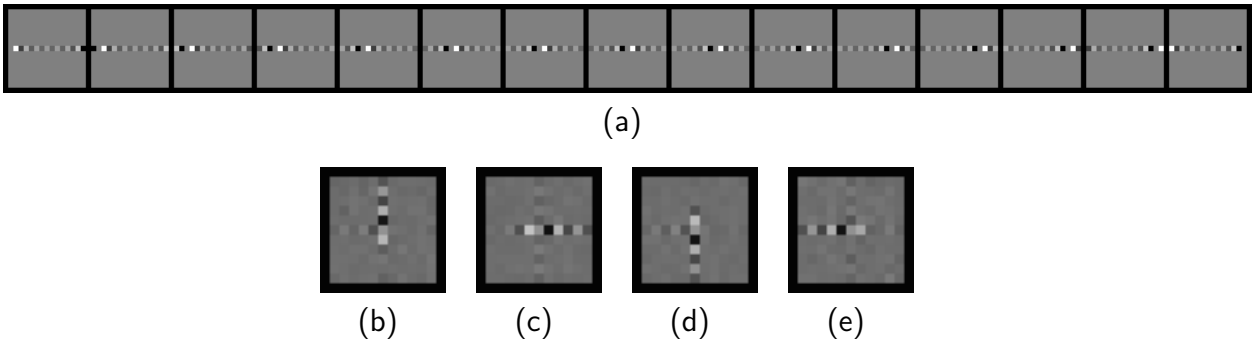


Figure 3.5: **Analytically specified and learned shift operators.** (a) The weights connecting the center row of output pixels to the input pixels. Each box in the grid displays weights for a single output pixel in frame $t+1$, arranged according to the spatial position of the connecting input pixel from frame t. This function effects horizontal translation with wrap-around boundary conditions. This operator and its vertical counterpart were used to generate data in the translation recovery task. (b-e) The weights connecting the center output pixel in frame $t+1$ to each input pixel in frame t, in each of the four $\mathbf{\Psi}_j$ operators. Note that in each learned function, the ringing pattern is one sided. The model has learned four operators, because each operator effects a shift operation with no boundary; that is, unlike the analytical operator pixels shifted off the patch by these operators do not come back in the other side of the patch. This is due to the masking operation used to remove the boundary condition in the data generation process.

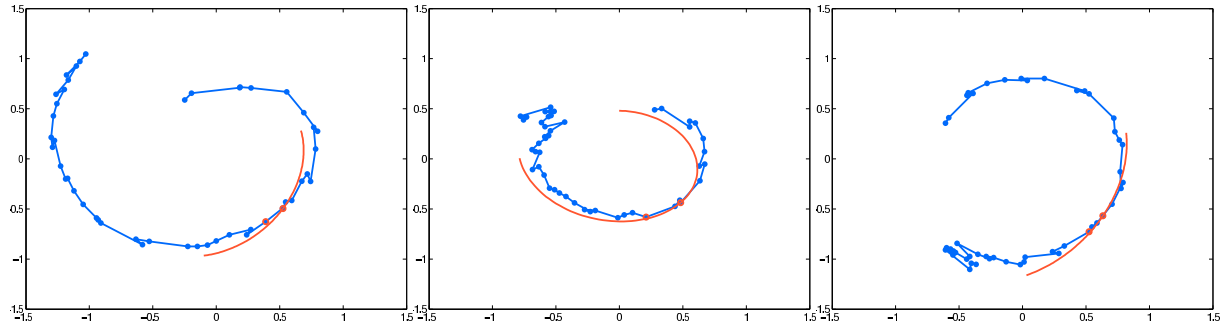


Figure 3.6: **Learning transformations of oriented filter pairs across time.** The orbits of three three complex filter outputs in response to a natural movie. The blue points denote the complex output for each frame in the movie sequence and are linked to their neighbors via the blue line. The points circled in red were observed by the model, and the red curve shows an extrapolation along the estimated trajectory.

image. A complex pyramid is built from each frame in a movie, and pairs of filter responses 1 to 4 frames apart are observed by the model. Four 2x2 basis functions are learned in the manner described above. Figure 3.6 shows three representative examples that illustrate how well the model is able to extrapolate from the solution estimated using the learned basis $\mathbf{\Psi}$, and complex responses from the same filter within a 4 frame time interval. In most cases, this trajectory follows the data closely for several frames.

We apply the model to natural movies using the binary window, PCA decomposition, and coarse-to-fine inference strategy described in Chapter 2. Rather than try to collapse the space of operators by adding a weight decay term to the energy function, we assume that the variety of transformations that occur between frames in natural movies are rich enough that we will run into computational limitations long before we provide enough operators to capture them all completely. Our energy function is:

$$E = \sum_t \frac{1}{2\sigma^2} \left|\left|\mathbf{x}(t+1) - \exp(\sum_j \mathbf{\Psi}_j \, c_j(t)) \, \mathbf{x}(t)\right|\right|_2^2 + \gamma \left|\left|\mathbf{c}(t)\right|\right|_1 . \tag{3.15}$$

We train on 10 frame sequences of frames, whitened and dimensionality reduced as described in Chapter 2. Two models were trained: one with $J = 9$ and the other with $J = 16$. Both models had $P = 15^2$, $L = 100$, and a binary mask along a one pixel border. The parameters learned in these models are shown in Figure 3.8, and contours of equal log-probability between the MAP estimated coefficients for two basis elements are shown in Figure 3.9. In contrast to the operators learned in the linear model of Chapter 2, none of the operators learned in the continuous transformation models have spatially uniform effects. Instead, they either

This figure must be viewed in a web browser.
Please view the effect of each transformtion basis element via the following links:
Ψ_1 Ψ_2 Ψ_3 Ψ_4 Ψ_5 Ψ_6 Ψ_7 Ψ_8 Ψ_9

Figure 3.7: **Exploring the effect of the learned Ψ_j operators.** The following movies show the effect of eachj Ψ_j basis function, as c_j is scanned across a range corresponding to its empirical distribution. The operator is applied to three different types of data, shown in four separate panels. Two of these panels are natural image patches. In the upper right panel are individual Fourier components, which have a regular structure that covers the entire patch, making it easier to understand the effect of the transformation. The lower left panels contains pink noise, which is colored by applying the unwhitening matrix to white Gaussian noise. Many of the basis elements cannot be characterized by a single type of effect, though there is often a dominant one. Ψ_2 seems to perform vertical translation and some deformation. Ψ_3 effects vertical translation in the upper part of the patch, and horizontal translation in the lower part. Ψ_4 effects horizontal translation and some rotation. Ψ_7 is dominated almost entirely by horizontal expansion. Ψ_8 performs horizontal translation and some rotation. Ψ_9 performs almost exclusively rotation.

become localized to a certain region, have an effect that gradually changes across space, or some combination of these two. The effect of these operators is easiest to visualize by applying them to a static natural scene and generating a movie; unfortunately, this is not easy to capture in paper form so Figure 3.7 links to movies of it that must be viewed in a web browser. It is clear that while translation motion has been learned, because it can be represented so compactly in just two operators, it does not dominate what they learn, which is a rich, diverse set of image distortions.

3.2 Factoring with continuous transformations

In Section 2.3, we introduced a general, two component form for factorization models, then explored specific choices for each component: a first order Taylor approximation for the transformation component, and a sparse coding model for the invariant image component. In this section we follow a similar progression, but explore different choices for each of the components. For the transformation component, we use the continuous transformation model described in Section 3.1, and introduce a new, group sparse coding model to describe the underlying invariant image.

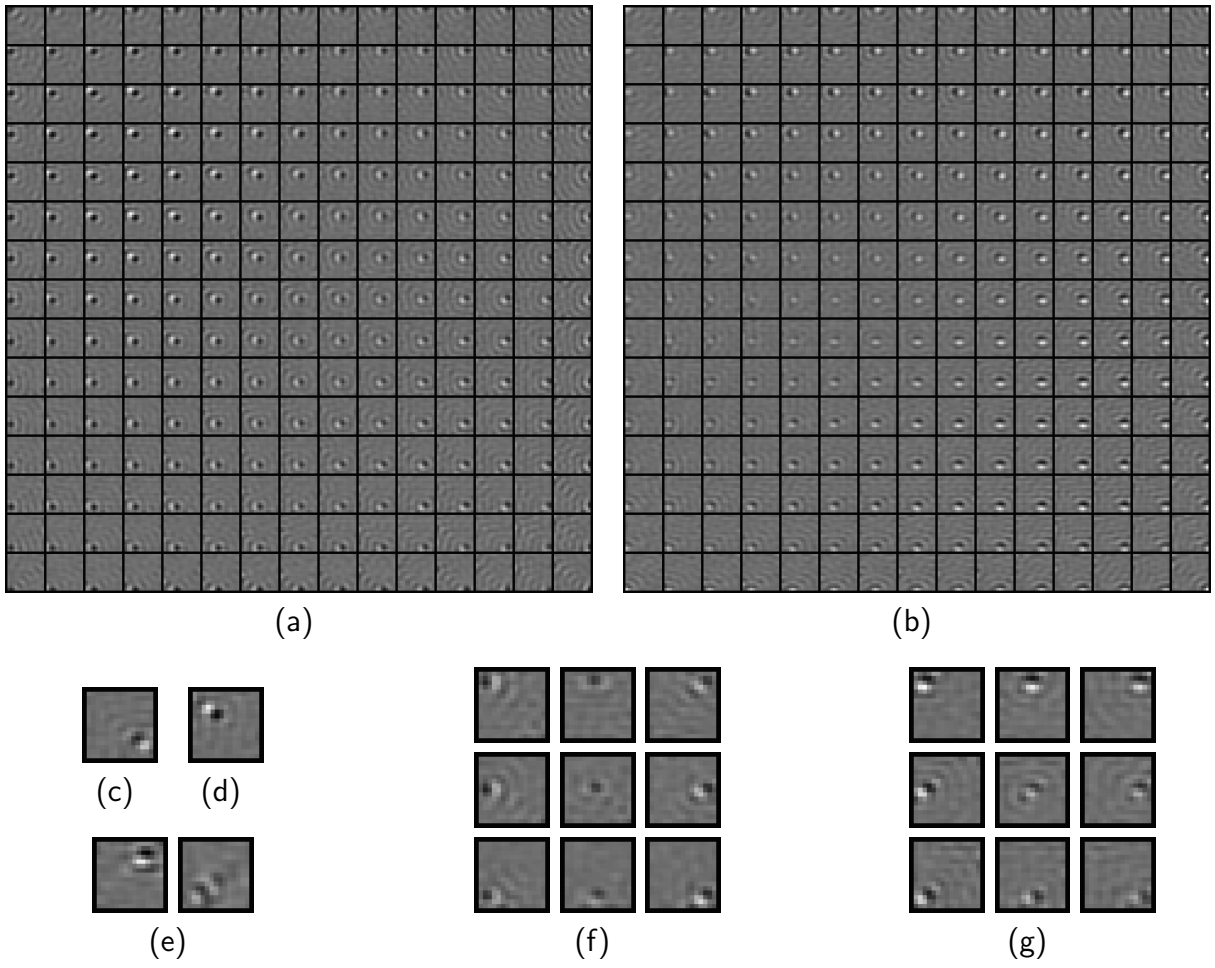


Figure 3.8: **Basis functions learned in a continuous transformation model with $J = 9$.** (a-b) Two of the nine continuous transformation operators learned from natural movies. Strikingly, not a single one of these nine learned operators effect the same transformation at each pixel: every function has a spatially varying pattern. (c-d) The weights for two strongly connected output pixels from two different basis elements. These are examples of a common form of weight patterns learned in many of the functions. (e) The weights for two output pixels taken from the same function with a large amount of variability across space to the transformation effected by it. The two shown output pixels are spatial derivatives but of different order and orientation. (f-g) Summaries of the spatial variability effects in two of the learned functions. (f) The nine sets of output pixel weights shown seem to produce an expansion or contraction, since the spatial derivatives flip orientation by 180 degrees across the center output pixel. (g) The nine sets of output pixel weights show seem to produce a rotation, as the orientation of the spatial derivative in each output pixel changes smoothly from the lower right to upper left corners of the output.

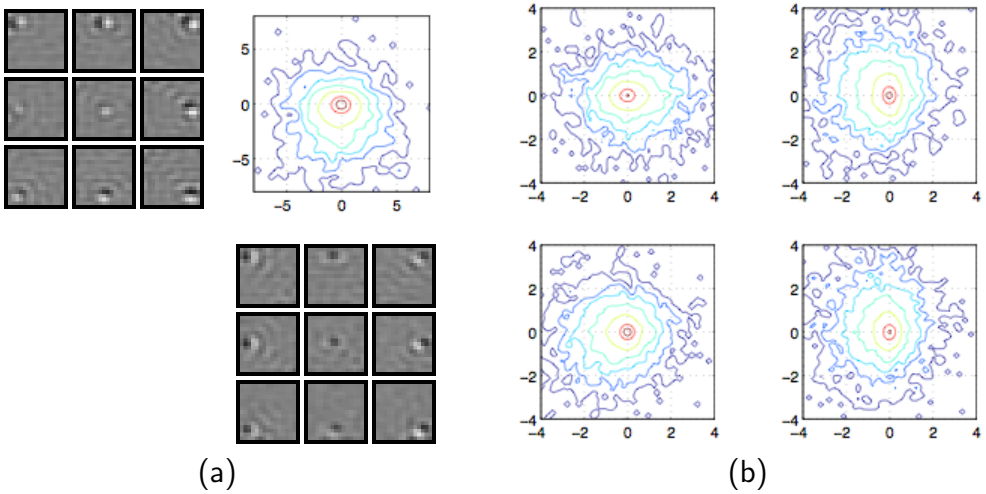


Figure 3.9: **Joint dependencies between continuous transformation basis elements.** (a) In a pair of functions taken from the model displayed in Figure 3.8, joint histograms of MAP estimates and their contours of equal log-probability are plotted. The prior (not shown) has diamond shaped equal log-probability contours, with the points of the diamond aligned to the axes. In contrast to the linear transformation model shown in Figure 2.2, here we see a joint distribution that is fairly close to the shape of the prior. We do not see any basis elements that have one-sided marginals: the coefficients for each basis element are approximately equally likely to be positive or negative. (b) Four joint histograms of coefficients from a randomly chosen subset of basis elements learned in a larger model, with $J = 16$. Nearly all pairs of coefficients have joint histograms that are roughly symmetric and factorial Laplacian. The histograms look more elliptically symmetric than diamond-shaped, which suggests that an elliptically symmetric sparse prior would be a better fit to the data.

3.2.1 Group sparse coding

Group sparsity refers to models in which the activities of groups of variables inhibit each other and themselves, but the singleton activities within the groups do not. The activity of a group is analogous to that of a complex cell in V1 – the activity of any cell with membership causes the group to become active. Group sparse coding is best understood as the following bilinear factorization model:

$$x_l = \sum_{jk} \Gamma_{ljk}\, c_{jk}\, d_j + n_l\,, \tag{3.16}$$

though by imposing some carefully chosen constraints, the parameters can be learned in an equivalent linear model. The bilinear model shown in Equation 3.16 has structure linking each d_j variable to a group of c_{jk} variables; the size of this group is K. Indices $j \in \{1, \ldots, J\}$ and $k \in \{1, \ldots, K\}$ address basis vectors across and within groups; thus setting $K = 2$, $J = 100$ would produce 100 groups with 2 elements each. For notational convenience, we denote a specific group of size K by $\mathbf{c}_j$, and also denote the k-th element of all J groups by $\mathbf{c}_k$. Some degrees of freedom are removed from the model by imposing positivity on the d_j variables, and angular structure on the c_{kj} variables. For $K = 2$, this angular structure is specified by these constraints:

$$c_{1j} = \sin(\theta_j) \tag{3.17}$$

$$c_{2j} = \cos(\theta_j)\,. \tag{3.18}$$

That is, both variables can be described by a single angle. We can generalize these constraints to larger K through the use of Euler angles, and the introduction of $K-1$ angular θ variables. The formulas given above denote the base case ($K = 2$) in a recurrence relation. To generalize the model to larger K, we use the following recurrence relation:

$$c_{kj}^{(K)} = \sin(\theta_{Kj})\, c_{kj}^{(K-1)} \qquad 0 < k < K \tag{3.19}$$

$$c_{kj}^{(K)} = \cos(\theta_{Kj}) \qquad k = K\,. \tag{3.20}$$

Under these constraints, each of the $\mathbf{c}_j$ vectors with K elements is generated from $K - 1$ angles, enforcing that for all settings of the underlying θ variables, each of the resulting $\mathbf{c}_j$ vectors lies on the surface of a K-sphere. Equivalently, the product of each $\mathbf{c}_j$ vector with its corresponding d_j can be viewed as a factorization of a zero-centered coordinate in a K-dimensional space into a set of $K - 1$ angles $\mathbf{c}_j$ and a non-negative radius d_j. Interestingly, this is a multi-angle generalization of the magnitude and phase factorization of [Cadieu and Olshausen, 2009]. By imposing a sparse prior on the magnitude variable only, we endow the model with the ability to learn relevant *subspaces* for coding the data, rather than just basis vectors. The subspaces that are selected to code a given data item should now give a coarse scale identification of its content, while the position within each subspace specifies finer level

details about its appearance. The energy function corresponding to this prior can be written concisely as follows:

$$E = ||\mathbf{x} - \sum_k \mathbf{\Gamma}_k \left(\mathbf{c}_k \odot \mathbf{d}\right)||_2^2 + \lambda \, ||\mathbf{d}||_1 \,, \tag{3.21}$$

where $\odot$ denotes element-wise multiplication. Although our derivation of this model began with the assumption that $\mathbf{x}$ would be a bilinear function of the $\mathbf{d}$ and $\mathbf{c}_j$ variables, due to the angular constraints we have chosen, there is a simpler, linear energy function that is equivalent and thus far easier to optimize:

$$E = ||\mathbf{x} - \sum_j \mathbf{\Gamma}_j \, \mathbf{a}_j||_2^2 + \lambda \sum_j ||\mathbf{a}_j||_2 \,. \tag{3.22}$$

where

$$a_{kj} = c_{kj} \, d_j \,. \tag{3.23}$$

Additionally, there is a simple equality that will recover the underlying $\mathbf{c}_j$ and $\mathbf{d}$ variables, given the $\mathbf{a}_j$ solution:

$$d_j = ||\mathbf{a}_j||_2 \tag{3.24}$$

$$c_{kj} = \frac{a_{kj}}{d_j} \,. \tag{3.25}$$

Note that c_{kj} is only defined if the magnitude of the group is non-zero. This form of sparsity is based on the model of [Hyvarinen and Hoyer, 2000] but also known in the statistics community as group Lasso [Yuan and Lin, 2006]. Under this convenient simplification, an image in our model $\mathbf{x} \in \mathbb{R}^L$ is generated from a linear super-position of vectors $\mathbf{z}_j$ plus i.i.d. Gaussian noise $\mathbf{n}$,

$$\mathbf{x} = \sum_j \mathbf{z}_j + \mathbf{n} \,, \tag{3.26}$$

where, each vector $\mathbf{z}_j$ is constrained to lie within a learned subspace; that is,

$$\mathbf{z}_j = \mathbf{\Gamma}_j \, \mathbf{a}_j \,. \tag{3.27}$$

There is a practical matter we must address before optimizing this model using MAP-EM. Due to the MAP approximation, there is a degeneracy in the learned solution: the model prefers to have all vectors in each subspace point in the same direction! The reason for this can be seen by considering the case where one vector in a subspace points in exactly the right direction as a data item we would like to reconstruct, but the others do not. In this case, the energy can always be reduced by distributing load from one coefficient onto many, and thus when only one element is used, the learning gradient will steer unused vectors in the subspace towards the vector that was used. Unfortunately, without maintaining the full posterior in

the E-step, this effect results in a positive feedback loop and every subspace collapses to point in a single direction. As is often the case when optimizing models with MAP-EM, we are forced to make a slight adjustment to our model to avoid this degeneracy. One solution is to interdigitate stochastic learning updates with an orthonormalization operation applied to each subspace [Hyvarinen and Hoyer, 2000], but this can be viewed as a projection onto a set of auxiliary constraints that are hidden from the gradient, and can potentially fight against it. Instead, we choose a convenient alternative energy that does not have this degeneracy: simply apply the i.i.d. Laplacian sparse prior to the norm of the contributing vectors $\mathbf{z}_j$, giving the following energy function:

$$E = \frac{1}{2}||\mathbf{x} - \sum_j \mathbf{z}_j||_2^2 + \lambda \sum_j ||\mathbf{z}_j||_2 \,. \tag{3.28}$$

Since the sparse prior is placed on the norm of the contributing vectors $\mathbf{z}_j$, there is no tendency for subspace vectors to align. Our model has a straightforward probabilistic interpretation:

$$-\log P(\mathbf{x}|\mathbf{z}) \propto \frac{1}{2}||\mathbf{x} - \sum_j \mathbf{z}_j||_2^2 \tag{3.29}$$

$$-\log P(\mathbf{z}_j) = -\log P(\mathbf{\Gamma}_j, \mathbf{a}_j) \propto \lambda \sum_j ||\mathbf{\Gamma}_j \, \mathbf{a}_j||_2 \,. \tag{3.30}$$

3.2.2 Learning phase transformations from image sequences

Composing the group sparse coding model with one continuous transformation model per group yields the following factorization model:

$$\mathbf{x}(t) = \sum_j \mathbf{\Gamma}_j \, \exp(\sum_m \mathbf{\Psi}_{mj} \, e_{mj}(t)) \, \mathbf{a}_j(t) + \mathbf{n}(t) \,. \tag{3.31}$$

We fit the model parameters to natural image sequences one component at a time. First, we learn a group sparse coding basis for natural image sequences by adding a stability prior to the energy function in Equation 3.28:

$$E = \sum_t \frac{1}{2}||\mathbf{x}(t) - \sum_j \mathbf{z}_j(t)||_2^2 + \lambda \sum_{jt} ||\mathbf{z}_j(t)||_2 + \gamma \sum_{jt} \dot{d}_j(t) \,. \tag{3.32}$$

and optimizing it using MAP-EM. The result is a set of parameters $\mathbf{\Gamma}_j$ that is optimized to natural image sequences, and produces the representation depicted in Figures 3.11 and 3.10(b). Second, an over-complete exponential model with weight decay is learned to compactly describe how the MAP-estimated coefficients within each subspace change over time. This

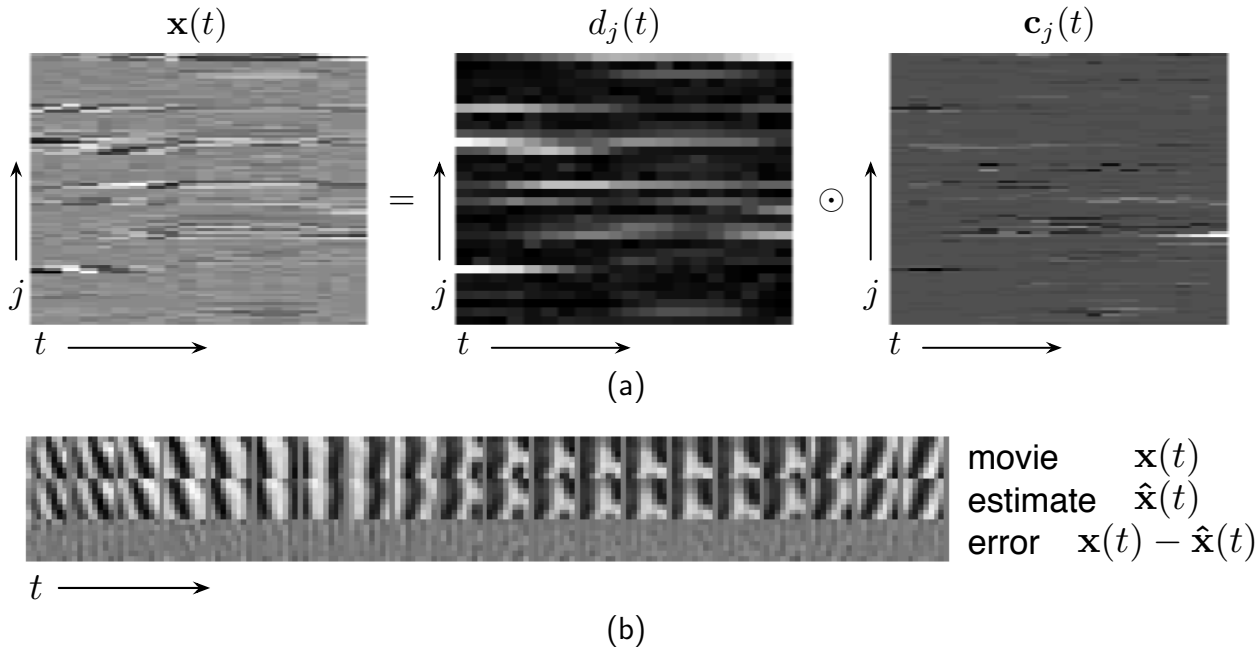


Figure 3.10: **A stabilized image sequence representation, using group sparse coding.** (a) After learning $\mathbf{\Gamma}_j$, inference is performed on a 20 frame sequence of an 8x8 pixel movie. The pixels $\mathbf{x}(t)$ (left) are encoded in terms of the product of a slow, non-negative amplitude $d_j(t)$ (center) and unconstrained time-varying phase coefficients $\mathbf{c}_j(t)$ (right). In this factorized form, the representation is sparse and temporally stable. (b) The MAP-estimated reconstruction of a 20 frame 8x8 pixel natural image sequence.

is accomplished in the framework described in Equation 3.1.1, with a weight decay prior. After convergence, 2-4 basis functions per subspace have significant norm. The resulting $\mathbf{\Psi}_j$ operators are depicted in Figure 3.10(a). The next step, which is to jointly estimate the $\mathbf{e}_j(t)$ and $\mathbf{a}_j(t)$ variables in Equation 3.31 using stability priors, is left for future work.

3.3 Discussion and concluding remarks

We have shown that it is possible to learn low-dimensional parameterizations of operators that transport along the non-linear manifolds formed by natural images over time. Surprisingly little work has previously been done on learning such high dimensional Lie groups. Early attempts to model the manifold structure of images train on densely sampled point clouds and find an embedding into a small number of coordinates along the manifold. However such an approach does not actually constitute a model, since there is no function for

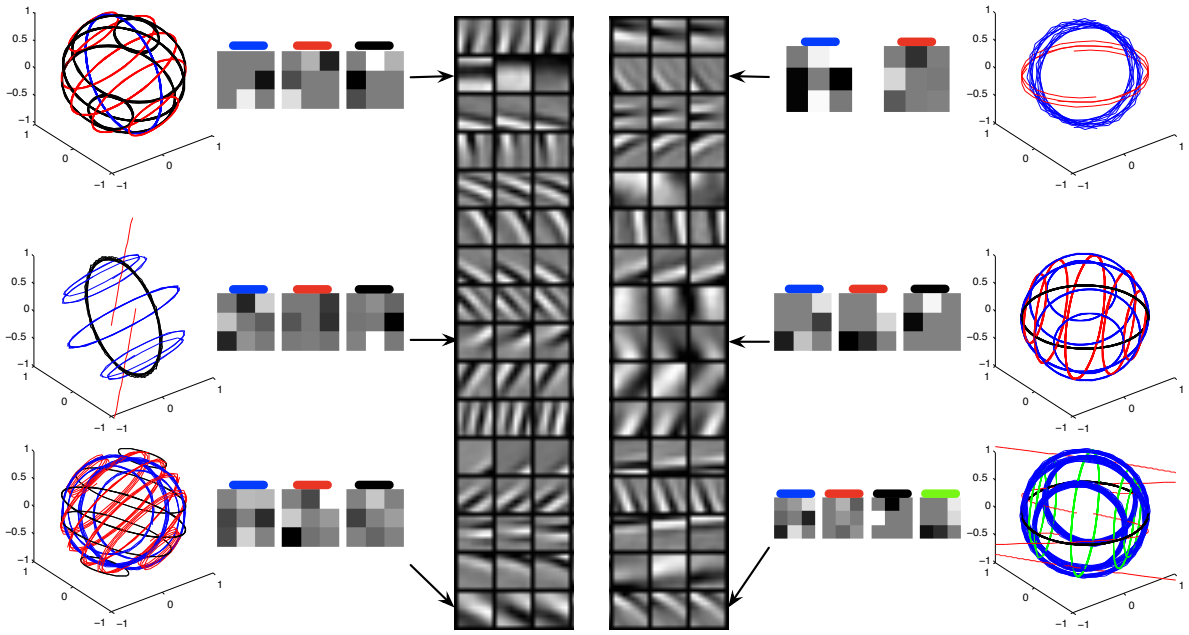


Figure 3.11: **A compact representation for coding phase changes among groups of basis functions.** At the far left and right of the figure are trajectories generated by a representative subset of the learned continuous transformation operators, using the 6 poles of the unit sphere as initial conditions, super-imposed on each other in 3-space. Each operator is shown in a different color, which corresponds to the color of the bar sitting above one 3x3 image adjacent to the trajectory. Each 3x3 image represents an operator, whose real values have been converted to gray-scale intensities, scaled by the maximum absolute value and symmetric around zero, which appears gray; positive values are lighter, and negative values are darker. The group of corresponding lower-level features whose coefficients follow each trajectory are in the row indicated by the arrow. The effect of a particular operator can be ascertained by animating the sum of these three lower-level features multiplied by their coefficients as the coefficients are modulated according to the trajectory. Most trajectories, when animated this way, result in the translation of an edge in a direction perpendicular to the orientation of the group. Prominent exceptions to this are the red trajectories of the second and sixth groups, which modulate the contrast of their group.

mapping arbitrary points, or moving along the manifold. One must always refer back to original data points on which the model was trained – i.e., it works as a lookup table rather than being an abstraction of the data. Here, by learning operators that transport along the manifold we have been able to learn a compact description of its structure.

An advantage of having such a model-based representation is that it can be leveraged to compute geodesics. One possible approach is to use a numerical approximation to the arc length integral:

$$S = \int_0^1 ||\mathbf{A}\, \exp(\mathbf{A}\, t)||_2^2 \, \mathrm{d}t = \lim_{T\to\infty} \sum_{t=1}^{T} || \exp(\mathbf{A}\, \frac{t}{T})\, \mathbf{x}(0) - \exp(\mathbf{A}\, \frac{t-1}{T})\, \mathbf{x}(0)||_2^2 \,, \qquad (3.33)$$

where T is the number of segments chosen to use in the piecewise linear approximation of the curve, and each term in the summation gives the length of a segment. We believe that this aspect of our model will be of use in difficult classification problems, such as face identification, where Euclidean distances measured in pixel-space give poor results.

Previous attempts to learn Lie group operators have focused on linear approximations. Here we show that utilizing the full Lie operator/matrix exponential in learning, while computationally intensive, is tractable, even in the extremely high dimensional cases required by models of natural movies. A spectral decomposition of $\mathbf{A}$ is the key component that enables this, and, in combination with careful mitigation of local minima in the objective function using a coarse-to-fine technique, gives us the power to factor out large transformations from data.

Appendix 3.A A circuit for MAP inference

Due to the fact that trajectories from the model can be extrapolated by integrating a linear ODE, our model has an elegant interpretation as a possible mechanism by which the cortex could make predictions about the world. Two circuits that operate on different time scales are necessary. On a slower scale – perhaps 10 Hz – a state change prediction is accumulated into an internal variable representing the predicted state. This circuit runs at a speed that is matched to the speed at which predictions about the world are needed for taking actions. On a faster scale, another circuit continuously descends an objective, optimizing the $\mathbf{c}$ variables to produce state change predictions about the world. This inner loop needs to converge before a new state of the world arrives. Specifically, what is *not* necessary is neural circuitry for computing matrix exponentials or eigen-decompositions of matrix exponents. For this reason, implementation of this system in a circuit would likely be strikingly computationally efficient compared to the way we have described its implementation on a general purpose computer.

The circuit shown in Figure 3.12(a) demonstrates the 'feed-forward' operation of the

system: given some internal representation $\mathbf{x}$ of the state of the world and a setting of the $\mathbf{c}$ variables that designates a direction of movement along the manifold at time t, the circuit produces a predicted set of changes to the internal state vector.

$$\hat{\dot{x}}_k(t) = \sum_{lj} \psi_{klj}\, c_j\, x_l(t) \tag{3.34}$$

These changes are accumulated in a second internal state vector which naturally decays, but is 'charged up' by continuously integrating the output of the feedforward circuit in a loop. Thus, this internal state vector maintains an exponentially decaying moving average of the predicted state, by integrating predicted state *changes*. The rate of decay should be matched to the statistics of the environment so that the window is not too large or small.

Figure 3.12(b) illustrates an interlocking circuit that runs on a time-scale faster than the frequency with which new states of the world are delivered, and compares the predicted state change with the observed change the world. By taking advantage of axonal, or electronically imposed propagation delays, the next observed state can become available in coincidence with the prediction. The circuit makes adjustments to the $\mathbf{c}$ variables, which modulate couplings between the observed and predicted states, to correct prediction errors. On a much slower time-scale than either of the two previous circuits, a similar third circuit (not shown) makes adjustments to the $\mathbf{\Psi}_j$ variables, which represent the manifold structure of states in the world.

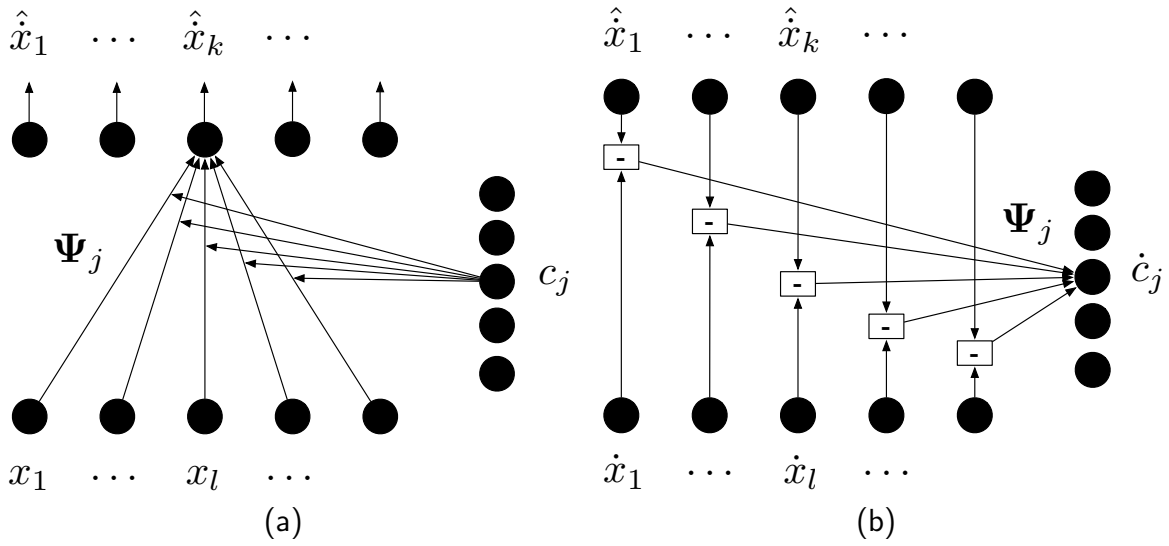


Figure 3.12: **A circuit for computing MAP estimates in the frame-to-frame continuous transformation model.** (a) Simulation of the continuous linear ODE predicts the next sensory input. (b) Inference of $\mathbf{c}$ variables is accomplished by comparing the predicted state change to the observed state change, weighted by $\boldsymbol{\Psi}_j$.

Chapter 4

Factoring with uncertainty

A perfect model for the image formation process in the natural world in combination with a rich prior that captures all of the regularities imposed by nature could be inverted with little need to represent uncertainty in the latent variables. That is, causes would be disambiguated with such certainty, that their posterior distributions would be well-represented by delta functions. In Chapters 2 and 3, this idea is essentially built in to our inference process, which represents posterior distributions by single samples taken at MAP estimates. However, we have by no means proposed a perfect forward model of the image formation process; in fact, we know at least one key assumption – a linear or bilinear image formation process – is wrong, because of the prevalence of occlusion in the real world. This knowledge that our models are only approximate motivates us to preserve uncertainty in full posterior distributions over our latent variables. To the extent that these distributions differ from delta functions, we know know there are flaws in our model; conversely, as our models improve we should observe a corresponding tightening of these posteriors towards delta functions. Thus, the preservation of uncertainty in posterior distributions provides a mechanism whereby we can measure incremental progress in a natural way during a sequence of proposed model changes.

This chapter anchors the main ideas presented in Chapters 2 and 3 in a directed probabilistic model. In Chapter 2 we introduced the idea of modeling short image sequences using a first-order ODE, and longer sequences through the use of time-varying function in the matrix exponent. Though the priors and learning schemes are both probabilistically motivated, the models described in the previous chapters lack some of the properties we desire in probabilistic models. For example, we would like to be able to measure the probability of the model given some data, synthesize image sequences without running a complicated sampling procedure, and visually compared image sequences synthesized from the model to the data. In this chapter we introduce a model and framework that meets all of these requirements. A straight-forward Monte Carlo algorithm for fitting the models is presented, which requires virtually no tuning or modification as extensions to the model are added and experimented with. First, we develop a bilinear model of static images. When fit to natural scenes, the

model parameters learn sensible group dependency structures, and we show that the likelihood of this bilinear model under hold-out data is superior to that of the mcRBM [Ranzato and Hinton, 2010]. Image patches generated from a closely related linear-exponential model exhibit the elongated structures, texture, and distribution of contrast that are characteristic of natural image patches. Next, we extend the linear-exponential model to be causal in time, developing ideas introduced in Chapter 2 and 3, but this time in a framework that preserves full posterior distributions over the latent variables. We demonstrate how modeling progress can be measured by evaluating the log likelihood of each model in this sequence of progressively more complex models, and, lastly, we propose an extension to time-varying images which links several static image models together using an auto-regressive process.

4.1 Factoring static images with uncertainty

We begin by showing how a sparse representation for static images may be factorized to learn higher-order features of natural scenes. We propose two layers of representation: the first layer is a sparse code; the second layer factors these sparse variables into two linear decompositions. Importantly, we show how Hamiltonian Markov chain Monte Carlo simulations may be used to sample from the posterior, enabling learning of the second layer. When trained on natural images, the first layer features learn oriented, localized bandpass functions and the second layer features learn structural dependencies that produce more elongated functions.

4.1.1 Bilinear model

Our first model of static images that preserves uncertainty in the posterior is a bilinear factorization model, closely related to group sparse coding of Section 3.2.1. Rather than having each basis function accompanied by one 'phase' variable, and imposing a fixed grouping structure by linking several phases to a common magnitude, in the following model the entire grouping structure is learned in the model parameters by maximizing log likelihood. The forward model is:

$$\mathbf{x} = \sum_j \mathbf{\Gamma}_j \, c_j \, \mathbf{d} + \mathbf{n} \,, \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^L$ is a vector of PCA whitened and dimensionality reduced pixels, each $\mathbf{\Gamma}_j \in \mathbb{R}^{L \times K}$ is a basis element for a linear operator, $\mathbf{n} \in \mathbb{R}^L$ is mean-zero i.i.d. Gaussian noise with variance σ^2, and the variables $\mathbf{c} \in \mathbb{R}^J$ and $\mathbf{d} \in \mathbb{R}^K_{\geq 0}$ encode a latent representation of

the image. The constraints that define our model's density are:

$$\mathbf{c} \sim \mathcal{N}(0, \mathbf{I}) \tag{4.2}$$

$$\mathbf{d} \sim \mathcal{E}(1) \tag{4.3}$$

$$\mathbf{x} \,|\, \mathbf{c}, \mathbf{d} \sim \mathcal{N}(\mathbf{\Phi}\,\mathbf{a}, \sigma^2\,\mathbf{I})\,, \tag{4.4}$$

where

$$\mathbf{a} = \mathbf{\Theta}\;\mathbf{c} \;\odot\; \mathbf{\Psi}\,\mathbf{d}\,. \tag{4.5}$$

$\mathcal{E}(1)$ denotes the exponential distribution with rate parameter one, and the operator $\odot$ performs an element-wise product. Here we have factorized the tensor $\mathbf{\Gamma}_j$ from Equation 4.1 into the product of three matrices,

$$\Gamma_{jkl} = \sum_m \Phi_{lm}\,\Psi_{mj}\,\Theta_{mk}\,, \tag{4.6}$$

reducing the model parameters from JKL to $M(J + K + L)$, where M is the number of factors coupling each l, j, and k. Factorizing tensors this way has recently been used successfully by [Memisevic and Hinton, 2010; Ranzato *et al.*, 2010a] to impose constraints that are reasonably matched to structure in the data being modeled.

This density can also be viewed as a linear generative model,

$$\mathbf{x} = \mathbf{\Phi}\,\mathbf{a} + \mathbf{n}\,, \tag{4.7}$$

with a non-factorial prior over $\mathbf{a}$, parameterized by $\mathbf{\Psi}$ and $\mathbf{\Theta}$. If $\mathbf{\Psi} = \mathbf{I}$ and $\mathbf{\Theta} = \mathbf{I}$, it takes the form of a Gaussian scale mixture [Wainwright and Simoncelli, 2000]. We depart from previous work by relaxing the assumption of a factorial distribution for $\mathbf{a}$. The intent is that $\mathbf{\Psi}$ learns cliques in $\mathbf{a}$ which tend to coactivate, and $\mathbf{\Theta}$ learns to enforce consistency between the elements in $\mathbf{a}$ by inducing, for example, sign agreement.

4.1.2 Model estimation

We wish to maximize the likelihood of our model given the data by estimating the parameters $\mathbf{\Lambda} \equiv \{\mathbf{\Phi}, \mathbf{\Psi}, \mathbf{\Theta}\}$. Since $\mathbf{c}$ and $\mathbf{d}$ are latent, we use Expectation Maximization (EM), which iteratively maximizes a lower bound on the log-likelihood, derived using Jensen's inequality:

$$\log P(\mathbf{x}|\mathbf{\Lambda}) \geq \int Q(\mathbf{c}, \mathbf{d}|\mathbf{x}) \log \frac{P(\mathbf{x}, \mathbf{c}, \mathbf{d}|\mathbf{\Lambda})}{Q(\mathbf{c}, \mathbf{d}|\mathbf{x})} d\mathbf{c}\, d\mathbf{d}\,. \tag{4.8}$$

The lower bound on the right hand side is called the expected complete log likelihood, $\mathcal{L}(Q, \mathbf{\Lambda})$. In the case that $Q(\mathbf{c}, \mathbf{d}|\mathbf{x}) = P(\mathbf{c}, \mathbf{d}|\mathbf{x}, \mathbf{\Lambda})$, $\mathcal{L}(Q, \mathbf{\Lambda})$ becomes the true log likelihood. Using the superscript t to distinguish the parameters at each of the iterates, we alternate

between maximizing $\mathcal{L}(Q, \mathbf{\Lambda})$ with respect to $\mathbf{\Lambda}$ (M-step),

$$\mathbf{\Lambda}^{(t+1)} = \arg\max_{\mathbf{\Lambda}} \mathcal{L}(Q^{(t+1)}, \mathbf{\Lambda}) \,, \tag{4.9}$$

and updating $Q(\mathbf{c}, \mathbf{d}|\mathbf{x})$ (E-step),

$$Q^{(t+1)}(\mathbf{c}, \mathbf{d}|\mathbf{x}) = P(\mathbf{c}, \mathbf{d}|\mathbf{x}, \mathbf{\Lambda}^{(t)}). \tag{4.10}$$

In the E-step we update samples from the distribution computed at the previous time step $Q^{(t)}(\mathbf{c}, \mathbf{d}|\mathbf{x})$ using a Hamiltonian Monte Carlo sampler, described in Appendix 4.A. In the M-step, we maximize the lower bound $\mathcal{L}(Q, \mathbf{\Lambda})$ with respect to $\mathbf{\Lambda}$, which amounts to maximizing

$$\int Q(\mathbf{c}, \mathbf{d}|\mathbf{x}) \log P(\mathbf{x}, \mathbf{c}, \mathbf{d}|\mathbf{\Lambda}) \, d\mathbf{c} \, d\mathbf{d} \,. \tag{4.11}$$

This is equivalent to minimizing the energy, E, averaged over the samples, since our density is factorial over data items and the normalization factor Z is constant:

$$\mathbf{\Lambda}^{(t+1)} = \arg\min_{\mathbf{\Lambda}} \frac{1}{B} \sum_{b=1}^{B} E(\mathbf{q}_b) \,. \tag{4.12}$$

We accomplish this using the L-BFGS algorithm [Byrd *et al.*, 1995].

4.1.3 Model recovery

Before applying the model to real datasets, we first show that our learning algorithm can recover model parameters used to generate artificial data. Some care must be exercised in the choice of parameters to recover, because many choices belong to a class of equivalent or near-equivalent solutions in the model parameters, making recovery impossible even when the learning algorithm is working perfectly. To state this more precisely, the model parameters are not identifiable, a fact that is true of almost every model that is sufficiently rich to capture structure in natural images. For this reason we make the following choices: $\mathbf{\Phi} = \mathbf{I}$, the entries of $\mathbf{\Theta}$ are drawn randomly from the Laplace distribution with scale parameter 1, and $\mathbf{\Psi}$ is set to have the following non-trivial, overlapping grouping structure: with half as many $\mathbf{d}$ variables as $\mathbf{c}$ variables, we couple each $\mathbf{d}$ variable to three outputs of $\mathbf{\Theta}\mathbf{c}$ by activating three entries of each $\mathbf{\Psi}$ column, and setting the rest to zero. In rows with two active elements, the active elements are set to $1/\sqrt{2}$, otherwise they are set to 1, giving the rows unit norm. The configurations used to generate data from the model are illustrated in the first column of Figure 4.1. We set σ to 0.1, L, M and J to 16, and K to 8, then generate 2,000 samples using Equation 4.7.

Next, we estimate the model using our algorithm, starting from random initialization.

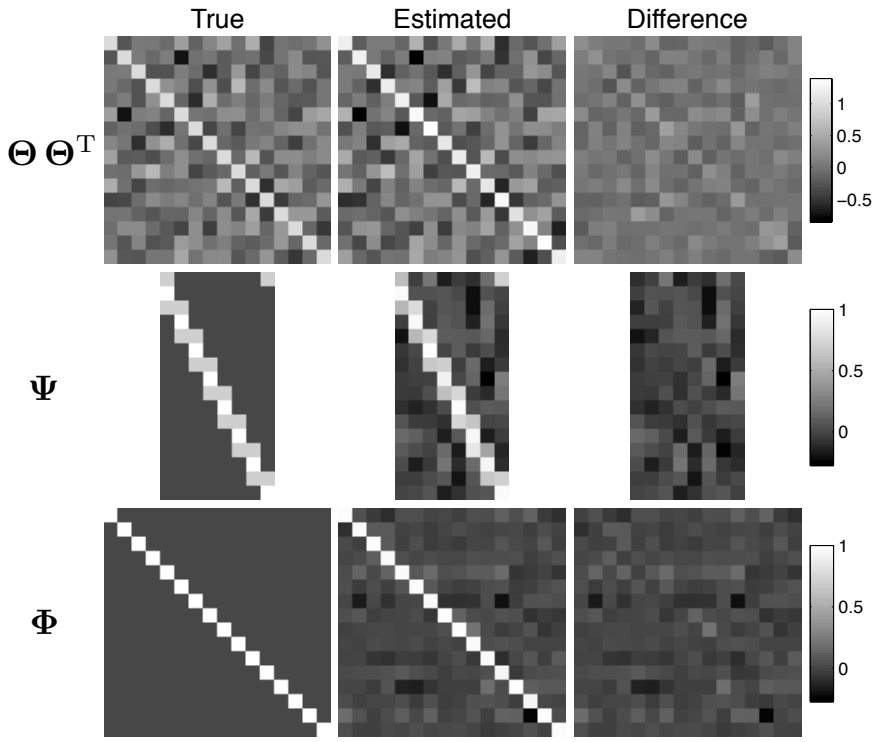


Figure 4.1: Model recovery. 2,000 data samples were generated from the model by setting the parameters as shown in the 'True' column, drawing from the priors for $\mathbf{c}$, $\mathbf{d}$, and $\mathbf{n}$, then generating $\mathbf{x}$ samples via Equation 4.7. The model parameters were then estimated by our E-M algorithm, from random initial conditions. The 'Estimated' column shows the parameters after 10,000 learning iterations. The 'Difference' column shows the difference between true and estimated parameters.

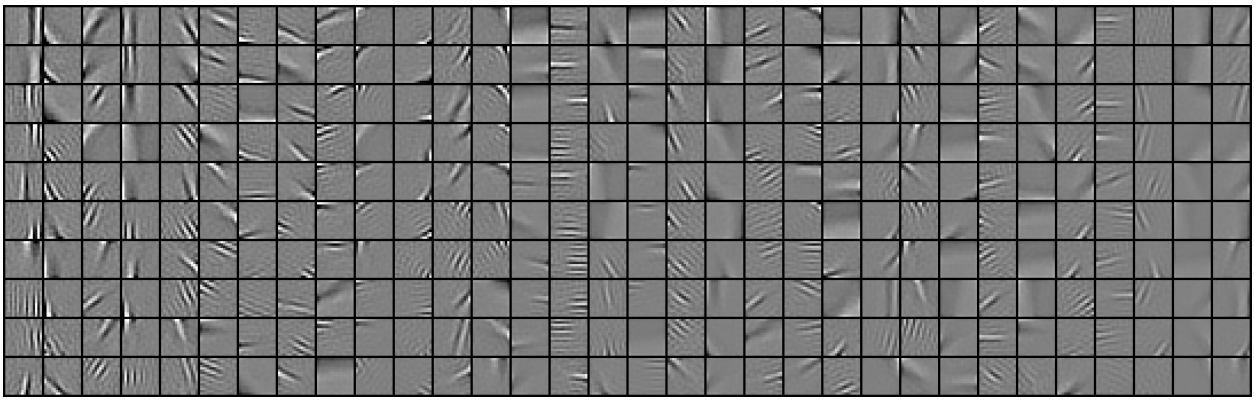
The columns of $\mathbf{\Phi}$ are set to random unit length vectors, $\mathbf{\Theta}$ to $\mathbf{I}$, and all the entries of $\mathbf{\Psi}$ are set to $1/\sqrt{K}$. Since there is a multiplicative degeneracy between the length of the columns of $\mathbf{\Phi}$ and the rows of $\mathbf{\Psi}$ and $\mathbf{\Theta}$, after every M-step we multiply by the appropriate factors to maintain unit norm in the columns of $\mathbf{\Phi}$ and the rows of $\mathbf{\Psi}$. This does not affect our algorithm in any way, but ensures that the estimated parameters will have comparable scales to those used to generate the data. Figure 4.1 shows the parameters recovered after 10,000 iterations of E-M in the 'Estimated' column. (Most of the correct structure is apparent after only 1,000 iterations.) Though imperfect, the estimation process does a remarkable job at recovering the known parameters for this non-trivial example.

4.1.4 Experiments on natural images

We train two bilinear models on 50,000 16x16 pixel image patches taken at random from 4,112 linearized images of natural scenes from the van Hateren dataset [van Hateren and van der Schaaf, 1998]. The extracted image patches are first logged, and then mean subtracted. They are then projected onto the top 128 PCA components, which retains 99.84% of the variance of the original patches, and whitened by rescaling each dimension to unit norm. For both models, $L = 128$ (the number of PCA components). The smaller model has $J = 144$, $K = 144$, and $M = 144$; the larger has $J = 256$, $K = 256$, $M = 256$. We set σ to 0.1, initialize $\mathbf{\Psi}$ and $\mathbf{\Theta}$ to $\mathbf{I}$, and $\mathbf{\Phi}$ to random with unit norm columns, then run 40,000 iterations of E-M using $\tau = 10$, $\epsilon = 0.01$, and 8 iterations of L-BFGS in the M-step, which takes about two days on an 8-core machine. We use the same procedure to train a linear generative model with a Laplace prior, for comparison. In this case, the only parameters are $\mathbf{\Phi} \in \mathbb{R}^{L \times M}$ and we use $M = 200$.

The learned parameters for the large bilinear model are shown in Figures 4.2 and 4.3 (the smaller model learns something qualitatively similar). Each column of Figure 4.2(a) shows 10 columns of $\mathbf{\Phi}$, organized into groups and sorted according to the strength of the corresponding weight in a column of $\mathbf{\Psi}$. The columns of $\mathbf{\Phi}$ learn to be oriented, bandpass functions. Interestingly, the spatial extent of each $\mathbf{\Phi}$ basis element shrinks when the marginal over $\mathbf{a}$ is allowed to be non-factorial. Since a single $\mathbf{d}$ variable can then control a whole group of functions, the model learns to link together edges with similar orientation to form elongated structures. This is evident in Figure 4.2(b), where the image component controlled by several of the $\mathbf{d}$ variables is shown to produce an elongated structure, and one that is significantly more complex than a Gabor. If $\mathbf{\Theta c}$ is set to be a vector of all ones, the averaged group activations shown in Figure 4.2(b) are the image components contributed by a single $\mathbf{d}$ variable. Notice there is significant variety among these groups, and that they capture several properties of natural image patches, such as elongated structures across the entire patch, and different types of oriented and non-oriented textures. These elongated structures are less apparent in the samples from the bilinear model, which suggests that the exponential prior activates too many $\mathbf{d}$ variables simultaneously. This could mean that a prior with more mass at zero would yield a better model.

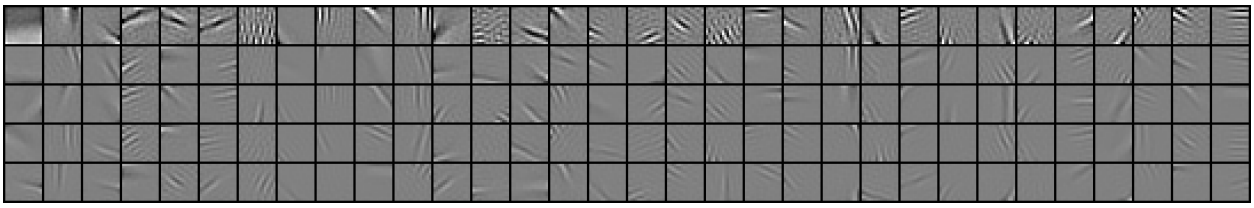
The coupling structure encoded by $\mathbf{\Theta}$ can be visualized by displaying the columns of $\mathbf{\Phi}$ that correspond to the largest entries in each column of $\mathbf{\Sigma} = \mathbf{\Theta}\mathbf{\Theta}^{\mathrm{T}}$, which gives the covariance of the variable $\mathbf{f} = \mathbf{\Theta c}$ – a useful way to interpret $\mathbf{\Theta}$. In Figure 4.3, each displayed column corresponds to a column in $\mathbf{\Sigma}$, with the upper squares holding the bases with the strongest corresponding $\mathbf{\Sigma}$ entries.



(a)

(b)

Figure 4.2: Dependencies between basis functions learned in $\mathbf{\Psi}$ from natural images. (a) The 10 most positively coupled columns in $\mathbf{\Phi}$ are shown for 32 randomly selected columns in $\mathbf{\Psi}$ (out of 256). Bases are weighted by their corresponding coupling strength in $\mathbf{\Psi}$. (b) The average value of all weighted bases. Note that these averages look strikingly similar to a natural image patch. The fact that these averages seem to look more like natural image patches than actual samples from the model suggests that the exponential prior on the $\mathbf{d}$ variables is not sparse enough.



(a)

(b)

Figure 4.3: Dependencies between basis functions learned in $\mathbf{\Theta}$ from natural images. (a) The 5 most strongly coupled columns in $\mathbf{\Phi}$ are shown for 32 randomly selected columns in $\mathbf{\Sigma}$ (out of 256). Bases are weighted by their corresponding coupling strength in $\mathbf{\Sigma}$. (b) The average value of all weighted bases. Note that, unlike for $\mathbf{\Psi}$, the bases combine constructively, suggesting that $\mathbf{\Theta}$ is enforcing consistency and sign agreement.

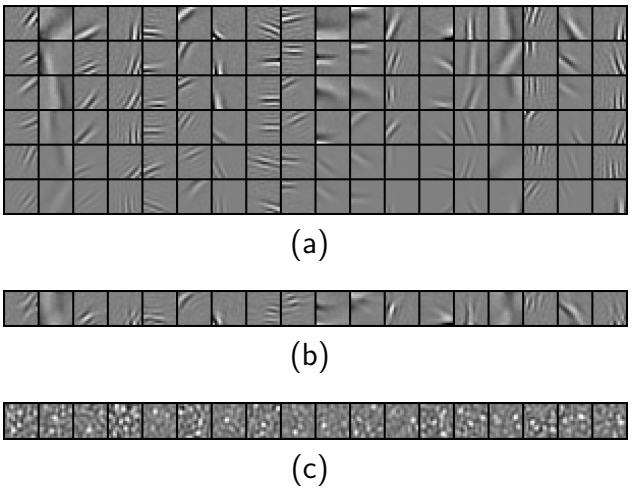


Figure 4.4: A subset of the learned parameters for the mcRBM (256) model. (a) Each of the 16 columns shows the receptive fields given by columns of the $\mathbf{C}$ matrix, connected through a column of $\mathbf{P}$, with the contrast scaled to reflect the strength of the connection. (b) The average of the receptive fields pooled by the columns of $\mathbf{P}$. (c) A subset of the columns of $\mathbf{W}$, the 'mean' units.

4.1.5 Comparison to mcRBM

The mean and covariance restricted Boltzmann machine (mcRBM) [Ranzato and Hinton, 2010] is an undirected analogue of our directed bilinear model. The high quality of the samples produced by the mcRBM has motivated the exploration of similar models [Courville *et al.*, 2010], and for these reasons it is a particularly relevant comparison. We train the mcRBM using the marginal energy function E_{mcRBM} as it was implemented in the released code:

$$E_{\text{mcRBM}}(\mathbf{x}) = -\sum_k \log\left(1 + \exp(\frac{1}{2}\sum_m P_{mk}\frac{(\mathbf{C}_m\,\mathbf{x})^2}{||\mathbf{x}||_2^2 + \frac{1}{2}} + b_k^{\text{c}})\right) - \sum_j \log\left(1 + \exp(\mathbf{W}_j\,\mathbf{x} + b_j^{\text{m}})\right) + \frac{1}{2\sigma^2}\mathbf{x}^{\text{T}}\mathbf{x} - \mathbf{x}^{\text{T}}\,\mathbf{b}^{\text{v}}\,, \tag{4.13}$$

with parameters: $P \in \mathbb{R}^{M\times K}$, $C \in \mathbb{R}^{M\times L}$, $W \in \mathbb{R}^{J\times L}$, $b^{\text{m}} \in \mathbb{R}^J$, $b^{\text{c}} \in \mathbb{R}^K$, $b^{\text{v}} \in \mathbb{R}^K$, $\sigma \in \mathbb{R}$. We train two mcRBM models, both with $L = 128$. For the smaller model we use $J = 144$, $K = 144$, $M = 144$; for the larger, $J = 256$, $K = 256$, $M = 256$. The learned parameters are shown in Figure 4.4. We follow the training protocol described in the paper exactly, save for one detail: instead of using CD-1, with a 20 leap-frog step Hamiltonian Monte Carlo

proposal distribution, we train using CD-5, with the same proposal distribution. Initially, we followed the training protocol exactly, but found that the resulting model assigned extremely high likelihoods to spurious 'pot holes' in the state space. In particular, low- to no-contrast patches were given likelihoods 50 orders of magnitude higher than patches with contrast. A symptom of this is that when trying to sample from the trained model at equilibrium, the sampler gets stuck in low-contrast states. The use of CD-5 ameliorates this behavior, but samples drawn from the models we learned using CD-5 do not match the contrast distribution of natural images as well as those drawn from models we learned using CD-1.

4.1.6 Linear-exponential model

In our bilinear model, the scale variable $\mathbf{d}$ followed an exponential distribution. This choice was made mainly for two reasons: it guarantees positivity, which ascribes some structure to its symbolic meaning, and has a heavy tailed shape, which is better matched to the statistics of natural scenes than a one-sided Gaussian or Chi distribution – we determined this empirically, through likelihood measurements. Another distribution that satisfies these properties is the log-Normal distribution. Such a variable can be obtained by exponentiating a normally distributed variable. Thus, by substituting $\exp(\mathbf{d})$ for $\mathbf{d}$ in our forward model for $\mathbf{a}$, and changing the prior over $\mathbf{d}$, this model is also easy to explore using our sampler. To determine the appropriate scale for the exponent, we also learn an addition set of parameters, $\mathbf{\Omega} \in \mathbb{R}^{K \times K}$, in the following model:

$$\mathbf{c} \sim \mathcal{N}(0, \mathbf{I}) \tag{4.14}$$

$$\mathbf{d} \sim \mathcal{N}(0, \mathbf{I}) \tag{4.15}$$

$$\mathbf{x} \,|\, \mathbf{c}, \mathbf{d} \sim \mathcal{N}(\mathbf{\Phi}\,\mathbf{a}, \sigma^2\,\mathbf{I})\,, \tag{4.16}$$

where

$$\mathbf{a} = \mathbf{\Theta}\ \mathbf{c}\ \odot\ \mathbf{\Psi}\ \exp(\mathbf{\Omega}\,\mathbf{d})\,. \tag{4.17}$$

4.1.7 Evaluation

Separately, we have developed a practical, robust method for log likelihood estimation that is based on Annealed Importance Sampling [Neal, 2001]. We use this technique to evaluate the log likelihood of the bilinear model, the linear-exponential model, and a few other relevant models. Table 4.1 gives this comparison, showing that our model is a significant (> 10 nat) improvement over the mcRBM. These results were obtained using a Gaussian proposal distribution in AIS, and 10^5 intermediate distributions in which the target and proposal distributions were mixed linearly. The numbers reported favor the mcRBM somewhat, because experiments using $10^{5.5}$ intermediate distributions in the smaller (144) models result in slightly higher likelihoods for the directed models, but do not change the likelihoods

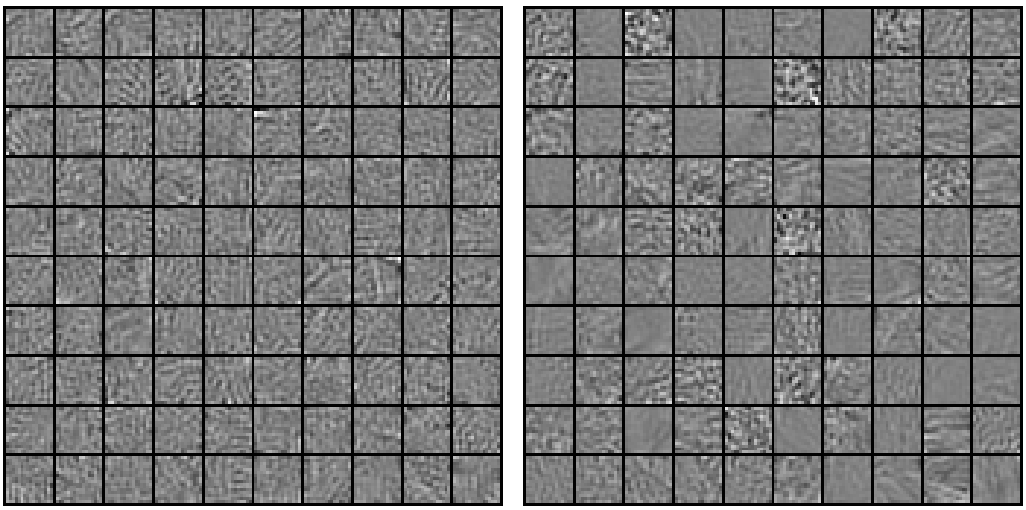
Model	Avg. log likelihood
directed linear, Laplace prior	-167.45 ± 8.19
mcRBM 144	-147.00 ± 8.44
mcRBM 256	-142.40 ± 8.31
directed bilinear 144	-136.12 ± 9.32
directed linear-exponential 144	-135.64 ± 9.68
directed bilinear 256	-132.02 ± 9.19

Table 4.1: **Model quality comparison.** Average log-likelihood is measured on a hold-out set of 100 image patches. Although the standard deviation of the average log-likelihood is high, the relative ordering is still significant because all models were evaluated on exactly the same image patches. The labels 144 and 256 refer to specific model configurations described in the main text.

significantly for the mcRBM. The reason for this asymmetry in the effect of a larger number of intermediate distributions is because for the mcRBM, AIS is used to estimate a single normalization constant, whereas in the directed models we use it to integrate out the latent variables for each test image. Thus, in the former case all test images contribute to a single estimate, and thus it becomes more accurate with fewer intermediate distributions; the latter case effectively requires an independent estimate for each test image. However, because the improvement is minor, does not change the relative ordering, and requires a significantly greater amount of computation, we report all results using 10^5 intermediate distributions.

It is important to note that our training scheme for the mcRBM may not have been optimal. While we have complete confidence in the accuracy of the results obtained using CD-5, the quality of the model would likely improve if we were to use a method specifically designed to eliminate the pot-holes we noticed, such as persistent contrastive-divergence (PCD) [Tieleman, 2008]. For example, in the convolutional version of the mcRBM [Ranzato *et al.*, 2010b], the authors use an alternate training scheme with persistent samples from the model [Tieleman and Hinton, 2009].

In Figure 4.5, we provide samples from our models for visual comparison. The most striking observation from the samples is that while the linear-exponential model is only slightly more likely than our original bilinear model, achieving -135.64 ± 9.68 nats, compared to the bilinear model's -136.12 ± 9.32, the samples generated from it look much more like natural image patches than those generated from even a much larger bilinear model. Thus, we have not sacrificed any measurable likelihood by making this change to the prior distribution over the $\mathbf{d}$ variables, and the samples have clearly improved, which is encouraging. Additionally, since all of the latent variables in the model now have Gaussian priors, it is easy to chain them together causally through time.



(a) Bilinear, factorial

(b) Bilinear

(c) Linear-exponential

(d) Natural image patches

Figure 4.5: A comparison of samples drawn from the learned models, shown in the original space after projecting through an unwhitening transform. (a) Bilinear model (256) with $\mathbf{\Psi}$ and $\mathbf{\Theta}$ locked to $\mathbf{I}$. Samples were drawn from the priors for $\mathbf{c}, \mathbf{d}$ and $\mathbf{n}$, then used to generate from the model according to Equation 4.7. (b) $\mathbf{\Psi}$ and $\mathbf{\Theta}$ were learned. (d) Linear-exponential model (144) in which $\mathbf{\Phi}, \mathbf{\Psi}, \mathbf{\Theta}$ and $\mathbf{\Omega}$ were learned. (c) Natural image patches from the data set. Note that the samples from the linear-exponential model have many similar properties to the actual image patches: the distribution of contrasts, and the existence of both elongated edge structures and rich textures covering the entire patch.

4.2 Factoring time-varying images with uncertainty

Having introduced a framework for preserving uncertainty in full posterior distributions over our latent variables in a model of static natural images, we now return to the subject of time. In Chapter 2, we achieved our goal of temporal stability by adding a term to our model's energy function of the form $||\dot{\mathbf{a}}||_2^2$. In this final section, we describe a different approach, which is to encode the goal of temporal stability in a way that maintains causality in time. In some sense, the difference between these two approaches is a bit philosophical, because both are valid ways of encoding probability distributions. However, going forward we prefer the directed approach because in this case, inference under the model has the straightforward interpretation as the recovery of causes. The main reason this is our goal is because we believe a representation that perfectly encodes the causes will be most parsimonious.

We derive our time-varying model by linking together chains of latent variables in time using an AR(1) process. Because all of the latent variables in the linear-exponential model are Gaussian, this is extremely straight-forward. The chain on the $\mathbf{c}(t)$ variables is given by:

$$\mathbf{c}(0) \sim \mathcal{N}(0, \lambda^2 \mathbf{I}) \tag{4.18}$$

$$\mathbf{c}(t) \,|\, \mathbf{c}(t-1) \sim \mathcal{N}(\lambda \, \mathbf{c}(t-1), \mathbf{I} \, (1 - \lambda^2)) \,. \tag{4.19}$$

Likewise, the chain on the $\mathbf{d}(t)$ variables is given by:

$$\mathbf{d}(0) \sim \mathcal{N}(0, \gamma^2 \mathbf{I}) \tag{4.20}$$

$$\mathbf{d}(t) \,|\, \mathbf{d}(t-1) \sim \mathcal{N}(\gamma \, \mathbf{d}(t-1), \mathbf{I} \, (1 - \gamma^2)) \,. \tag{4.21}$$

These AR(1) processes have been parameterized such that they have marginal covariance $\mathbf{I}$. The parameters λ and γ must be chosen such that $|\lambda| < 1$ and $|\gamma| < 1$. Values closer to one encourage temporal stability in the posterior distributions over $\mathbf{c}$ and $\mathbf{d}$. The observation process is essentially as it was in the static image case:

$$\mathbf{x}(t) \,|\, \mathbf{c}(t), \mathbf{d}(t) \sim \mathcal{N}(\mathbf{\Phi} \, \mathbf{a}(t), \sigma^2 \, \mathbf{I}) \,, \tag{4.22}$$

where

$$\mathbf{a}(t) = \mathbf{\Theta} \, \mathbf{c}(t) \, \odot \, \mathbf{\Psi} \, \exp(\mathbf{\Omega} \, \mathbf{d}(t)) \,. \tag{4.23}$$

Our goal in this model is to replace the undirected stability constraint from the previous chapters with one that is causal in time, and we stop short of exploring this in combination with a continuous transformation model – this will be a direction of future work.

4.3 Discussion and concluding remarks

We have introduced a flexible, hierarchical bilinear model and a straight-forward Hamiltonian Monte Carlo algorithm for learning its parameters. In contrast to MAP-EM (used throughout Chapters 2 and 3), which collapses posteriors to delta functions, this algorithm allows us to preserve uncertainty in the posterior distributions over latent variables. Preserving uncertainty in this way reflects the fact that we do not believe our forward models for image formation are perfect, and so a given image should not necessarily induce a particular representation in the model, but rather a hedged bet over possible representations that illuminates those that are most probable.

We have shown that when optimized, our bilinear model learns groupings of oriented, localized, bandpass features. While the constituent members of each group are related in these properties, there is considerable overlap between groups: many lower-level features participate in several groups, in contrast to the structure prescribed by group sparse coding (Section 3.2.1). Performing inference in this model factors an image into two distinct types of causes: group activations, and phase relationships between the lower-level features in each group. An important distinction between this type of factorization and that which is described in Chapters 2 and 3 is that here we have not yet captured a description of the time-varying transformation that maps between an underlying invariant and its visual appearance in the scene.

Although it is achieved in a simplistic sense, without the benefit of learning parameterized representations of frame to frame transformations, this type of two component factorization is achieved in our subsequent linear-exponential model for time-varying images. The main goal of this model is to achieve perceptual stability in the latent representation, in combination with causality in time and the preservation of uncertainty, and it succeeds in that regard. Adding learned transformation representations to this framework, and measuring their improvements to the log-likelihood of the model will be a direction of future work.

Appendix 4.A Hamiltonian Monte Carlo

In order to sample $P(\mathbf{c}, \mathbf{d}|\mathbf{x}, \mathbf{\Lambda})$, we use a variation on a Langevin dynamics sampler with partial momentum refreshment. The underlying technique was introduced in [Horowitz, 1991], and is clearly presented in [Neal, 2010] (Sections 5.2 and 5.3). We combine the variables we wish to sample from as $\mathbf{q} \equiv \{\mathbf{c}, \mathbf{d}\}$, and refer to $\mathbf{q}$ as our 'position' variables. We introduce auxiliary 'momentum' variables $\mathbf{p} \sim \mathcal{N}(0, \mathbf{I})$ – one for each $\mathbf{q}$. The distribution of interest can be recovered by sampling from the joint distribution over $\mathbf{q}$ and $\mathbf{p}$,

$$P(\mathbf{q}, \mathbf{p}|\mathbf{x}, \mathbf{\Lambda}) = P(\mathbf{q}|\mathbf{x}, \mathbf{\Lambda})\, P(\mathbf{p})\,, \tag{4.24}$$

and then marginalizing over $\mathbf{q}$,

$$P(\mathbf{q}|\mathbf{x}, \mathbf{\Lambda}) = \int d\mathbf{p}\, P(\mathbf{q}, \mathbf{p}|\mathbf{x}, \mathbf{\Lambda})\,. \tag{4.25}$$

We sample from the joint distribution $P(\mathbf{q}, \mathbf{p}|\mathbf{x}, \mathbf{\Lambda})$ by alternating between simple updates for the momentum variables, and Metropolis updates in which a new state is proposed by computing a trajectory according to Hamiltonian dynamics. The simple momentum update is

$$\mathbf{p}' = -\sqrt{1-\beta}\,\mathbf{p} + \sqrt{\beta}\,\mathbf{r}, \tag{4.26}$$

where $\mathbf{r} \sim \mathcal{N}(0, \mathbf{I})$, and $\beta \in [0, 1]$ is a randomization rate. If $\mathbf{p}$ is a univariate Gaussian, then $\mathbf{p}'$ will also be a univariate Gaussian, as required. We set β so as to randomize half of the momentum power per unit simulation time under the Hamiltonian dynamics, allowing the samples to traverse extended paths before their momentum is replaced. If ϵ is the duration of a single simulation step, the operation in Equation 4.26 is repeated $T = \frac{1}{\epsilon}$ times per unit simulation time. We set the fraction of the original power remaining after T steps to $\frac{1}{2}$, and solve for β,

$$\beta = 1 - \exp\left(\epsilon \log\left(1/2\right)\right)\,. \tag{4.27}$$

The Hamiltonian for the system is the negative log likelihood of the joint distribution over $\mathbf{q}$ and $\mathbf{p}$,

$$H(\mathbf{q}, \mathbf{p}) = E(\mathbf{q}) + \mathbf{p}^{\mathrm{T}}\,\mathbf{p} + \log Z\,, \tag{4.28}$$

where Z is a normalization constant. $E(\mathbf{q})$ is the contribution from $P(\mathbf{q}|\mathbf{x}, \mathbf{\Lambda})$,

$$E\left(\mathbf{q}\right) = \frac{1}{2\sigma^2}||\mathbf{n}||_2^2 + \frac{1}{2}||\mathbf{c}||_2^2 + \mathbf{1}^{\mathrm{T}}\mathbf{d}\,, \tag{4.29}$$

where the noise $\mathbf{n}$ is defined in Equation 4.7, $\mathbf{d} \geq 0$, and $\mathbf{1}$ is the vector of all ones. The dynamics associated with the Hamiltonian are

$$\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}} \qquad\qquad \dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}}\,. \tag{4.30}$$

We numerically approximate these dynamics for a time ϵ using a single step of a leapfrog

integrator (followed by negation of $\mathbf{p}$ so the update is reversible),

$$\mathbf{p}(t+\frac{\epsilon}{2}) = \mathbf{p}(t) - \frac{\epsilon}{2}\frac{\partial E}{\partial \mathbf{q}}(\mathbf{q}(t)) \tag{4.31}$$

$$\mathbf{q}(t+\epsilon) = \mathbf{q}(t) + \epsilon\,\mathbf{p}(t+\frac{\epsilon}{2}) \tag{4.32}$$

$$-\mathbf{p}(t+\epsilon) = \mathbf{p}(t\frac{\epsilon}{2}) - \frac{\epsilon}{2}\frac{\partial E}{\partial \mathbf{q}}(\mathbf{q}(t+\epsilon))\,, \tag{4.33}$$

which is particularly well suited to sampling scenarios since the update step is reversible and symplectic. Typically, the new position $\mathbf{q}(t+\epsilon)$ and momenta $\mathbf{p}(t+\epsilon)$ are then accepted or rejected based upon a standard Metropolis-Hastings criteria,

$$P_{\text{accept}} = \min[1, \exp\left(H(\mathbf{q}(t), \mathbf{p}(t)) - H(\mathbf{q}(t+\epsilon), \mathbf{p}(t+\epsilon))\right)]\,.$$

If the state is rejected, the position and momentum are reset to their value at the beginning of the trajectory. If the dynamics in Equation 4.30 are followed perfectly, $P_{\text{accept}} = 1$. Rejection therefore compensates for discretization errors during integration. Sampling is thus performed by repeating the following steps τ times:

1. Update the momentum $\mathbf{p}$ using Equation 4.26.
2. Update the position $\mathbf{q}$ and momentum $\mathbf{p}$ using Equations 4.31-4.33.
3. Use Equation 4.34 to either accept the update from step 2, or revert $\mathbf{q}$ and $\mathbf{p}$ to their state after step 1.

Rather than sampling from the posterior repeatedly for each item, we load an entire batch of data into memory, and evolve one particle for each data item simultaneously. This allows us to avoid a significant number of burn-in steps that would be required if we initialized particles with randomized positions and momenta on each iteration of E-M. The sampling algorithm is initialized once, at the beginning of learning, by loading a large set of data $\{\mathbf{x}_b\}_{b=1..B}$, allocating space for one particle (represented by a position and momentum) per data item, and initializing $\{\mathbf{q}, \mathbf{p}\}$ by drawing from $\mathcal{N}(0, \mathbf{I})$. The sampling time-step ϵ and the number of sampling iterations per learning step τ are chosen at the beginning of learning and held fixed. In our experiments we set $\epsilon = 0.01$ – which is small enough to cause only a slight discretization error – and never check M-H or reject. Since we measure the likelihood of the solutions we obtain, we know from experiments that this approximation does not reduce their quality, and in fact decreases the number of EM iterations required.

Since the exponential distribution is defined only over the positive real interval, there remains the practical matter of handling the constraint at zero, which we accomplish by negating the position and momentum for $\mathbf{d}$ variables when their position becomes negative after running a step in the dynamics, as described in [Neal, 2010] (Section 5.1 and Figure 8).

Chapter 5

Conclusion

The main idea put forth in this thesis is that invariances can be learned from the statistics of the visual world by fitting a two-component model to natural movies. In the abstract, the first component is a representation of an underlying invariant that serves to contain all information about an object that is necessary to generate its visual appearance. The second component represents the configuration of the object and its environment, and transforms the first component to produce the object's appearance according to natural viewing circumstances. An important difference from the way some other researchers have approached this problem is that in my approach, both of these model components have latent variables that explain each other away. That is, the latent variables cannot be inferred by a mechanism that computes in two separate, independent processing streams. To emphasize this point, consider the following visual question: "Is the appearance due to the illuminate or the shape of the object?" Because the solution for one latent variable will change the solution for the other, they *must* interact.

In Chapter 2, I gave a specific embodiment of this idea in the framework of probabilistic bilinear models. From the footings of a forward model for image sequences that is based on the solution to a linear ODE, I derived a two-component model that captures the idea of a continuously transformed underlying image invariant. As a start, this continuous transformation model was approximated with a first order Taylor expansion, which I fit to natural image sequences by learning model parameters that capture statistical regularities in how these underlying invariants are transformed during natural viewing. From the learned solution, it is evident that a more compact representation would be attained were the first order Taylor approximation forgone; this hypothesis is shown to be correct in Chapter 3. In Chapter 4, I construct a learning framework wherein full posterior distributions over the causes of an image sequence are maintained, and develop a new sequence model that is causal in time.

5.1 Summary of results

The main result from Chapters 2 and 3 is that expansions and contractions, rotations, and other complicated distortions that occur in the dynamics of movies can be captured in learned model parameters, and because they represent more precise motion priors than have previously been proposed, these models have the potential to achieve gains in video coding and optical flow estimation. They also succeed in providing stabilized, more symbolic representations of movies that will likely be useful in further, higher-level vision computations. In Chapter 4, my main result is image patches that the proposed linear-exponential model of static images generate, which exhibit many of the structural properties of natural images. Additionally, I provide empirical evidence – likelihood measurements – indicating that the linear-exponential model of static images is a better model of natural image patches than the previously most competitive contending model trained using contrastive divergence.

5.2 Future directions

Integrating a continuous transformation component into the hierarchical factorization model described in Chapter 4, yielding a two-component model, is the next most logical extension to pursue in the future. Because training transformation models on videos has always been extremely computationally intensive, I would also like to investigate several approaches for speeding it up:

1. Reduce the number of parameters in the basis tensor $\mathbf{\Psi}_j$ used in Chapters 2 and 3. One obvious way to do this is as a factored decomposition using three matrices, as done in the static image model described in Chapter 4. Since I can compare likelihoods, I will know exactly how much is sacrificed by the factored approximation to the full tensor. However, it would be better to use a representation that leverages the structure that has been learned. One way to do this would be to generate each of the rows of each $\mathbf{\Psi}_j$ matrix from a common basis.

2. Stop using E-M. Instead, simply treat all of the model parameters as latent variables, and sample them. I have begun doing this in static image models and it seems to result in significantly faster learning, especially at the beginning. Also, it simplifies the learning algorithm considerably.

3. Reduce the number of particles the sampler is required to evolve. One way to do this that I have been experimenting with recently in models of static images is to train using image sequences from movies, and evolve in the sampler only a single particle per movie patch sequence, instead of one particle per movie patch frame. Similar to learning with 'mini-batches' of data, where a partial M-step is computed from only a subset of all the data each learning iteration, the premise of this idea is that the posterior distribution

implied by a frame from a sequence is highly similar to the posterior implied by its sequential neighbor. Thus, after exchanging a frame for its neighbor, the particle representing a sample from the posterior can return to equilibrium after only a few integration steps of the Hamiltonian.

Though they have not been included in this document because I wanted it to focus exclusively on models of natural time-varying images, I have also applied many of these models to classification tasks, with favorable results. Especially in the context of small numbers of labelled training examples, learned parametric models that describe the transformation from an underlying invariant to a particular exemplar show promise, and I would like to develop these ideas further. One aspect that is currently lacking is a fully probabilistic classification mechanism that outperforms the results I am currently able to obtain by inferring MAP estimates which are then used in a support vector machine. I would like to better understand what is lacking from the forward model that prevents the same level of performance from being attained by simply inverting it.

The existence of the software framework I developed over the course of my studies [1] makes it markedly easier to leverage the ideas I have presented in the context of specific tasks that are typically used to measure progress in the field. Using my motion model to estimate optical flow seems like a promising direction to pursue, and I would like to show that the deformable motion models I have developed are able to characterize the distributions of motions in natural movies more precisely, and over much larger spatial extents than the smoothness priors built in to most optical flow methods. However, the most advanced probabilistic models for optical flow incorporate other advantageous details that my model lacks – such as a layered model of occlusion – and thus it might make more sense to integrate my motion model into an existing method. Another idea is to go in the opposite direction, and integrate a layered occlusion model into my own framework. Expressive power of the model could then be freed to learn additional structure in the data. An occlusion model would also implicitly provide a symbolic representation of depth, which is an important grouping cue.

One last direction that I have wanted to pursue since the beginning, which has really only now been made possible using the sampler described in Chapter 4, is to stack the model and attempt to learn to factor (again) a factored representation like the one depicted in Figure 2.9. Since factorization models are non-linear functions of their latent variables even in forward (generative) mode, the representation learned by stacking two factorization models on top of each other would likely capture richer, more interesting types of invariants. However, there are many details to be worked out in terms of exactly how to structure such a hierarchy.

[1] Available in my github repository.

Bibliography

[Ackley *et al.*, 1985] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

[Arathorn, 2002] D. W. Arathorn. *Map-Seeking Circuits in Visual Cognition: A Computational Mechanism for Biological and Machine Vision*. Stanford University Press, 2002.

[Baddeley, 1996] R. Baddeley. An efficient code in V1? *Nature*, 381:560–561, 1996.

[Baraniuk *et al.*, 2010] R. G. Baraniuk, V. Cevher, and M. B. Wakin. Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective. *Proceedings of the IEEE*, 2010.

[Barrow and Tenenbaum, 1981] H. G. Barrow and J. M. Tenenbaum. Computational vision. *Proceedings of the IEEE*, 69(5):572–595, 1981.

[Belkin and Niyogi, 2002] M. Belkin and P. Niyogi. Laplcian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, volume 14. Cambridge, MA: MIT Press, 2002.

[Bell and Sejnowski, 1997] A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[Bengio and Monperrus, 2005] Y. Bengio and M. Monperrus. Non-local manifold tangent learning. In *Advances in Neural Information Processing Systems*, volume 17, pages 129–136. Cambridge, MA: MIT Press, 2005.

[Bengio *et al.*, 2006] Y. Bengio, M. Monperrus, and H. Larochelle. Nonlocal estimation of manifold structure. *Neural Computation*, 18(10):2509–2528, 2006.

[Black and Anandan, 1996] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[Blake and Zisserman, 1987] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[Boureau *et al.*, 2010] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[Bregler and Malik, 1998] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8–15, 1998.

[Brox and Malik, 2011] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[Byrd *et al.*, 1995] R. H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

[Cadieu and Olshausen, 2009] C. F. Cadieu and B. A. Olshausen. Learning transformational invariants from natural movies. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21. MIT Press, 2009.

[Carlsson *et al.*, 2007] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *Int. J. Comput. Vision*, 76(1):1–12, 2007.

[Chen *et al.*, 1998] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20, 1998.

[Courville *et al.*, 2010] A. Courville, J. Bergstra, and Y. Bengio. Modeling natural image covariance with a spike and slab restricted Boltzmann machine. In *NIPS Workshop on Deep Learning*, 2010.

[Csurka *et al.*, 2004] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV*, pages 1–22, 2004.

[Dollar *et al.*, 2007] P. Dollar, V. Rabaud, and S. Belongie. Non-isometric manifold learning: Analysis and an algorithm. In *ICML*, pages 241–248, 2007.

[Efron *et al.*, 2004] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2], pages = 407–409), 2004.

[Einhauser *et al.*, 2002] W. Einhauser, C. Kayser, and K. P. Kording. Learning multiple feature representations from natural image sequences. In *International Conference on Artificial Neural Networks (ICANN)*, pages 21–26, August 2002.

[Figueiredo *et al.*, 2007] M. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[Foldiak, 1991] P. Foldiak. Learning invariance from transformation sequences. *Neural Comp.*, 3:193–199, 1991.

[Freeman and Tenenbaum, 1997] W. T. Freeman and J. B. Tenenbaum. Learning bilinear models for two-factor problems in vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 1997.

[Frey and Jojic, 1999] B. Frey and N. Jojic. Estimating mixture models of images and inferring spatial transformations using the em algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 416–422, 1999.

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

[Gabbiani *et al.*, 2002] F. Gabbiani, H. G. Krapp, C. Koch, and G. Laurent. Multiplicative computation in a visual neuron sensitive to looming. *Nature*, 420:320–324, 2002.

[Garrigues and Olshausen, 2007] P. Garrigues and B. Olshausen. Learning horizontal connections in a sparse coding model of natural images. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, 2007.

[Garrigues and Olshausen, 2010] P. Garrigues and B. Olshausen. Group sparse coding with a Laplacian scale mixture prior. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23, 2010.

[Glazer, 1987] F. Glazer. *Hierarchical Motion Detection*. Univ. of Massachusetts, Amherst, MA, 1987.

[Grimes and Rao, 2002] D. B. Grimes and R. P. N. Rao. A bilinear model for sparse coding. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15. MIT Press, 2002.

[Grimes and Rao, 2005] D. Grimes and R. Rao. Bilinear sparse coding for invariant vision. *Neural Computation*, 17(1):47–73, 2005.

[Hinton and Lang, 1985] G. E. Hinton and K. J. Lang. Shape recognition and illusory conjunctions. In *IJCAI*, volume 2, pages 252–259, 1985.

[Hinton, 1981] G. E. Hinton. A parallel computation that assigns canonical object-based frames of reference. In *IJCAI*, volume 2, 1981.

[Hofstadter, 1985] D. Hofstadter. *Metamagical themas.* Basic Books, 1985.

[Horn and Schunk, 1981] B. K. P. Horn and B. G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[Horowitz, 1991] A Horowitz. A generalized guided monte carlo algorithm. *Physics letters B*, Jan 1991.

[Hubel and Wiesel, 1962] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 73:218–226, 1962.

[Hyvarinen and Hoyer, 2000] A. Hyvarinen and P. O. Hoyer. Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12(7):1705–1720, 2000.

[Hyvarinen *et al.*, 2003] A. Hyvarinen, J. Hurri, and J. Vayrynen. Bubbles: A unifying framework for low-level statistical properties of natural image sequences. *Journal of the Optical Society of America*, 20(7):1237–1252, 2003.

[Irani and Anandan, 1999] M. Irani and P. Anandan. About direct methods. In *ICCV Workshop on Vision Algorithms*, pages 267–277, 1999.

[Jepson and Black, 1993] A. Jepson and M. Black. Mixture models for optical flow computation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 760–761, 1993.

[Karklin and Lewicki, 2003] Y. Karklin and M. S. Lewicki. Learning higher-order structures in natural images. *Network: Computation in Neural Systems*, 14:483–499, 2003.

[Karklin and Lewicki, 2006] Y. Karklin and M. S. Lewicki. Is early vision optimized for extracting higher-order dependencies? In *Advances in Neural Information Processing Systems (NIPS)*, volume 18, pages 625–642. MIT Press, 2006.

[Kavukcuoglu *et al.*, 2010] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[Kim *et al.*, 2007] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, 2007.

[Koenderink and van Doorn, 1997] J. J. Koenderink and A. J. van Doorn. The generic bilinear calibration-estimation problem. *International Journal of Computer Vision*, 23(3):217–234, 1997.

[Köster and Hyvärinen, 2007] U. Köster and A. Hyvärinen. A two-layer ica-like model estimated by score matching. In *ICANN*, 2007.

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

[LeCun *et al.*, 2004] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[Liu and Nocedal, 1989] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

[Lowe, 1999] D. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, 2:1150–1157, 1999.

[Lowe, 2004] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[Mairal *et al.*, 2008a] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, 2008.

[Mairal *et al.*, 2008b] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008.

[Memisevic and Hinton, 2007] R. Memisevic and G. E. Hinton. Unsupervised learning of image transformations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[Memisevic and Hinton, 2010] R. Memisevic and G. E. Hinton. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, 22(6):1473–1492, June 2010.

[Miao and Rao, 2007] X. Miao and R. P. N. Rao. Learning the lie groups of visual invariance. *Neural Computation*, 19(10):2665–2693, 2007.

[Miller *et al.*, 2000] E. G. Miller, N. E. Matsakis, and Paul A. Viola. Learning from one example through shared densities on transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 464–471, 2000.

[Neal, 2001] Radford Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, Jan 2001.

[Neal, 2010] Radford Neal. *MCMC using Hamiltonian dynamics*, chapter 5. Chapman & Hall, 2010.

[Olshausen and Field, 1996] 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.

[Olshausen and Field, 1997] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

[Olshausen *et al.*, 1993] B. A. Olshausen, C. H. Anderson, and D. C. van Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *The Journal of Neuroscience*, 13(11):4700–4719, 1993.

[Olshausen *et al.*, 2007] B. A. Olshausen, C. F. Cadieu, B. J. Culpepper, and D. K. Warland. Bilinear models of natural images. In *SPIE Proceedings: Human Vision and Electronic Imaging XII*, volume 6492, 2007.

[Olshausen, 2003] B. A. Olshausen. Learning sparse, overcomplete representations of time-varying natural images. In *IEEE International Conference on Image Processing*, September 2003.

[Ortiz *et al.*, 2001] M. Ortiz, R. A. Radovitzky, and E. A. Repetto. The computation of the exponential and logarithmic mappings and their first and second linearizations. *International Journal for Numerical Methods in Engineering*, 52:1431–1441, 2001.

[Osindero *et al.*, 2006] S. Osindero, M. Welling, and G. E. Hinton. Topographic product models applied to natural scene statistics. *Neural Computation*, pages 381–414, 2006.

[Raina *et al.*, 2007] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML 2007*, 2007.

[Ranzato and Hinton, 2010] Marc'Aurelio Ranzato and Geoffrey E Hinton. Modeling pixel means and covariances using factorized third-order boltzmann machines. *IEEE Conference on Computer Vision and Pattern Recognition*, Jan 2010.

[Ranzato *et al.*, 2007] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[Ranzato *et al.*, 2010a] M. Ranzato, A. Krizhevsky, and G. E. Hinton. Factored 3-way restricted Boltzmann machines for modeling natural images. In *AISTATS*, volume 13, 2010.

[Ranzato *et al.*, 2010b] M. Ranzato, V. Mnih, and G. Hinton. Generating more realistic images using gated mrf's. *Neural Information Processing Systems*, 2010.

[Rao and Ruderman, 1999] R. P. N. Rao and D. L. Ruderman. Learning lie groups for invariant visual perception. In *Advances in Neural Information Processing Systems (NIPS)*, volume 11, pages 810–816. MIT Press, 1999.

[Reichardt, 1969] W. Reichardt. *Processing of optical data by organisms and by machines*. Academic Press, 1969.

[Renninger and Malik, 2004] L. W. Renninger and J. Malik. When is scene recognition just texture recognition? *Vision Research*, 44:2301–2311, 2004.

[Roth and Black, 2007] S. Roth and M. J. Black. On the spatial statistics of optical flow. *International Journal of Computer Vision*, 74(3):33–50, 2007.

[Roweis and Saul, 2000] S.T. Roweis and L.K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

[Rozell *et al.*, 2008] C.J. Rozell, D.H. Johnson, R.G. Baraniuk, and B.A. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563, 2008.

[Simard *et al.*, 2001] P. Y. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. *International Journal of Imaging Systems and Technology*, 11(3), 2001.

[Sun *et al.*, 2008] D. Sun, S. Roth, J. P. Lewis, and M. J. Black. Learning optical flow. In *ECCV*, pages 83–97, 2008.

[Swindale, 1996] N. Swindale. Visual cortex: Looking into a klein bottle. *Current Biology*, 6(7):776–779, 1996.

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

[Tenenbaum *et al.*, 2000] J.B. Tenenbaum, V. Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

[Tibshirani, 1996] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, pages 267–288, 1996.

[Tieleman and Hinton, 2009] T. Tieleman and G.E. Hinton. Using fast weights to improve persistent contrastive divergence. *International Conference on Machine Learning*, 2009.

[Tieleman, 2008] T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *ICML*, pages 1064–1071, 2008.

[Tomasi and Kanade, 1992] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[Tompkins, 1941] C. Tompkins. A flat klein bottle isometrically embedded in euclidean 4-space. *Bulletin of the American Mathematical Society*, 47(6):508, 1941.

[Ungerleider and Mishkin, 1982] L. G. Ungerleider and M. Mishkin. *Two cortical visual systems*. MIT Press, 1982.

[van Hateren and van der Schaaf, 1998] J H van Hateren and A van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 265(1394):359–366, Jan 1998.

[VanGool *et al.*, 1995] L. VanGool, T. Moons, E. Pauwels, and A. Oosterlinck. Vision and lie's approach to invariance. *Image and Vision Computing*, 13(4):259–277, 1995.

[Wainwright and Simoncelli, 2000] M. J. Wainwright and E. P. Simoncelli. Scale mixtures of Gaussians and the statistics of natural images. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12, 2000.

[Wallis and Rolls, 1997] G. Wallis and E. T. Rolls. Invariant face and object recognition in the visual system. *Prog Neurobiol.*, 51(2):167–94, 1997.

[Weinberger and Saul, 2004] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Computer Vision and Pattern Recognition*, 2004.

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

[Woodham, 1980] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[Yang *et al.*, 2009] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[Yang *et al.*, 2010] J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[Yuan and Lin, 2006] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. R. Statistical Soc.*, 68(B):49–67, 2006.

[Zemel and Sejnowski, 1998] R.S. Zemel and T.J. Sejnowski. A model for encoding multiple object motions and self-motion in area MST of primate visual cortex. *Journal of Neuroscience*, 18(1):531, 1998.