Title
Learning Transformations From Video

Author
Wang, Ching Ming

Publication Date
2010

Peer reviewed|Thesis/dissertation

Learning Transformations From Video

by

Jimmy Ching Ming Wang

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

Doctor of Philosophy

in

Vision Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Bruno A. Olshausen, Chair
Professor Marty S. Banks
Professor David R. Brillinger

Fall 2010

Learning Transformations From Video

by

Jimmy Ching Ming Wang

Abstract

Learning Transformations From Video

by

Jimmy Ching Ming Wang

Doctor of Philosophy in Vision Science

University of California, Berkeley

Professor Bruno A. Olshausen, Chair

Our survival depends on accurate understanding of the environment around us through sensory inputs. One way to achieve this is to build models of the surrounding environment that are able to provide explanations of the data. Statistical models such as PCA, ICA and sparse coding attempt to do so by exploiting the second- and higher-order structures of sensory data. While these models have been shown to reveal key properties of the mammalian sensory system and have been successfully applied in various engineering applications, one shared weakness of these models is that they assume each observation is independent. In reality, there is often a transformational relationship between sensory data observations. Exploiting this relationship allows us to tease apart the causes of the data and reason about the environment. In this thesis, I developed an unsupervised learning framework that attempts to find the translational relationship between data and infer the causes of the observed data.

This dissertation is divided into three chapters. First, I propose an unsupervised learning framework that is able to model the transformations between data points using a continuous transformation model. I highlight the difficulties faced by previous attempts using similar models. I overcome these hurdles by proposing a learning rule that is able to compute the learning updates for an exponential model in polynomial time. I also propose an adaptive inference algorithm that is able to avoid local minima. These improvements make learning transformation possible and efficient.

Second, I perform a detailed analysis of the proposed model. I show that the adaptive inference algorithm is able to simultaneously recover multiple transformation parameters with high accuracy when given synthetic data where the transformation is known. When learned on pairs of images containing affine transformations, the algorithm correctly recovers the transformation operators. The unsupervised learning algorithm is able to discover transformations such as translation, illumination adjustment, contrast enhancement and lo-

cal deformations when learned on pairs of natural movie frames. I also show that the learned models provide a better description of the underlying transformation both qualitatively and quantitatively compare to commonly used motion models.

Third, I describe a plausible application for the continuous transformation model in video coding. In a hybrid coding scheme, I propose to replace the traditionally used exhaustive search motion model with transformation models learned on natural time-varying images. A detailed analysis of the rate distortion characteristics of different learned models is documented and I show that the learned model improves the performance of traditional motion models in various settings.

Acknowledgements

First and foremost I wish to thank my advisor Bruno Olshausen, who has spent countless hours mentoring me. I am truly inspired by his enthusiasm in research, his breadth of knowledge, and his ability to think about a problem deeply and thoroughly. Without him this thesis would not be possible. I thank my thesis committee members Marty and David their patience, support and encouragement.

I thank Jashua Sohl-Dickstein for a fruitful and productive collaboration. Jascha's creativity and his ability to comprehend abstract concepts is evident everywhere in this thesis. I owe him a great deal for tirelessly checking my derivatives and debugging my codes. I thank Ivana Tosic for our collaboration on video coding. Her insight in video compression has made this a much better thesis.

I thank the entire Redwood center for Theoretical Neuroscience for all the lively discussion we have had. Much of my inspiration in research is a product of interacting with members in the center. Special thanks to Jack Culpepper, Charles Cadieu, Vivienne Ming, Tim Blanch, Tony Bell, Urs Koster, Pierre Garrigues and Amir Khosrowshahi for their great friendship over the last five years.

Finally, I would like to thank my friends in the Vision Science program. Grandma, Mom and Dad, thank you for your loving support and advice through all these years.

Contents

Chapter 1

Introduction

1.1 Motivation

Our survival depends on being able to accurately understand and reason about our surroundings. Many researchers have suggested that our brain carries out these functions by learning a model of the environment that is consistent with sensory inputs, such as auditory and visual signals. A fundamental question here is how would one learn such a model? One possibility is discussed by Horace Barlow, where he argued that unsupervised learning should exploit the redundancy in data. Without redundancy it would be impossible to find any pattern or features in the data and in this sense **redundancy provides knowledge** [Barlow, 1989]. In this context, the redundancy in data is a result of the highly structured environment we live in. For example, in images of the visual world, there exists a considerable amount of regularity among pixel values across both space and time. To illustrate this fact, the two-point correlation function of natural images is plotted in figure 1.1 as a function of separation in space and time.

A simple model of the sensory input should capture such correlational structure. Principal Component Analysis (PCA) is an example of a model that does so. PCA represents signals using a set of orthonormal bases with decorrelated coefficients. Alternatively, it assumes that the data can be modeled with a multivariate Gaussian. When applied to natural images, instead of having the variances spreading over the entire signal, the variance is concentrated in a few principal components as shown in figure 1.2.

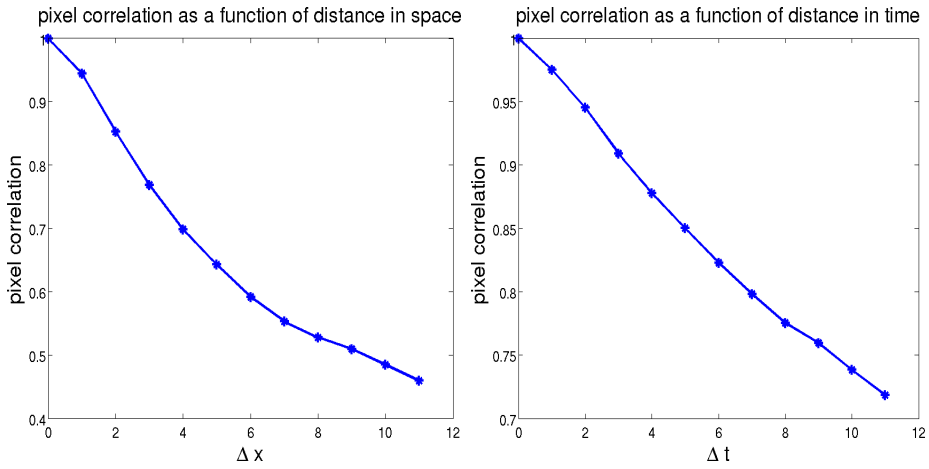

Figure 1.1: The two-point correlation function of natural images plotted as a function of separation in space (left) and time (right). 10000 pixel samples are used to generate the plots.

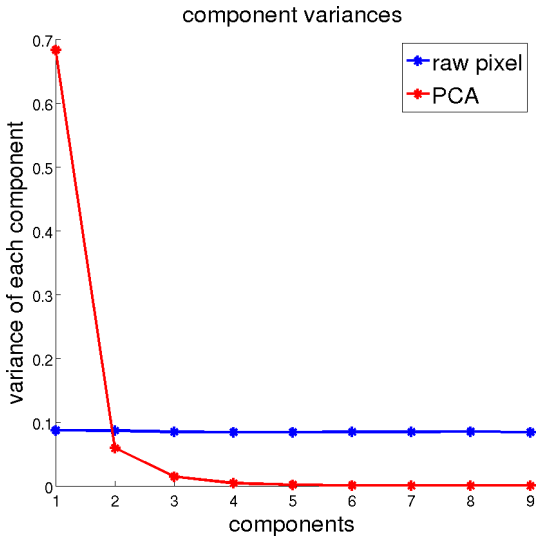

Figure 1.2: A comparison of the distribution of variances between PCA coefficients (red) and raw pixel values (blue). 1000 9-pixel natural image arrays are used to compute the PCA components and variance.

As George Box famously states "Essentially, all models are wrong, but some are useful." Although we know that real world data contains higher-order statistics, PCA is optimal for dimensional reduction in the least-square sense and has inspired image compression algorithms such as JPEG, where images are divided into smaller blocks and each block is

transform coded with the Discrete Cosine Transform (DCT), which approximates the principal components of natural images. Instead of recording all the pixel values, only the few significant (usually largest 15%) DCT coefficients are needed to capture the image content.

Taking one step forward, one can model higher-order statistics of the sensory data. Doing so requires moving away from the Gaussian assumption. In fact, [Friedman *et al.*, 1974] argues that basis functions whose projections are Gaussians are the most 'uninteresting' ones, and proposed to find basis functions whose projections are most non-Gaussian. Independent Component Analysis (ICA) [Bell and Sejnowski, 1997] and Sparse Coding [Olshausen and Field, 1996] attack this problem by describing the data as a multi-variate Laplacian distribution and are able to describe higher order statistics in data. The difference between PCA and ICA/sparse coding is depicted in figure 1.3, where a two-dimensional source (e.g. two-pixel images) are decomposed by both ICA and PCA. As shown, when the signals are non-Gaussian, ICA/sparse coding correctly identifies the components of some signals even when PCA does not.

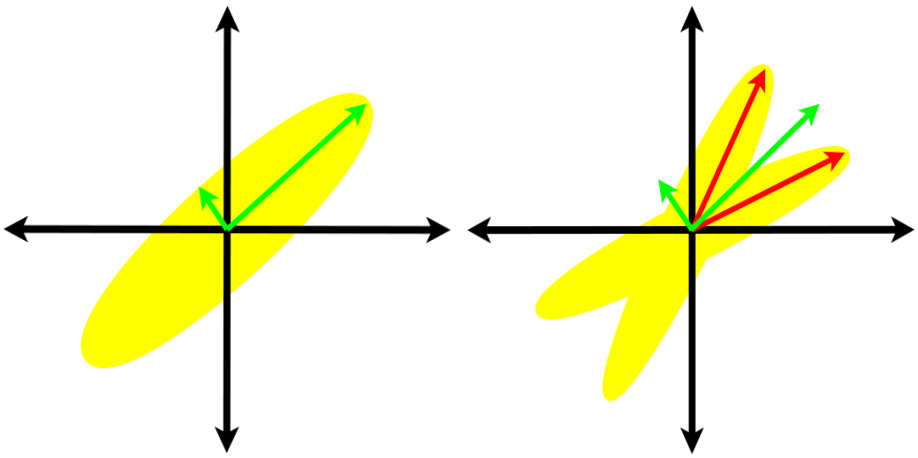

Figure 1.3: The figure on the left shows the two principal components of a two-dimensional signal that is distributed according to a Gaussian distribution. The figure on the right shows a non-Gaussian distribution. ICA (red) correctly identifies the signal components while PCA (green) does not.

By exploiting higher order structures, Olshausen and Field [Olshausen and Field, 1996] showed that basis functions learned using an unsupervised learning algorithm on natural images resemble receptive field structures found in early visual cortex. Smith and Lewicki [Smith and Lewicki, 2006] were able to show similar findings when learning from natural sound. This provides a strong clue that sparse coding could be a principled way to represent

static signals. In addition, it provides justification for the use of wavelets in image compression, which has been shown to provide a more compact description of images than DCT.

So far we have described linear models that account for second and higher-order statistics in static signals. However, the methods mentioned above consider each signal instance independently and ignore the relationships between signal instances. In natural signals, there is often a transformational relationship between signal instances. As a simple example, consider the set of three-dimensional signals formed by translating a three-pixel template by different amounts, shown in figure 1.4.

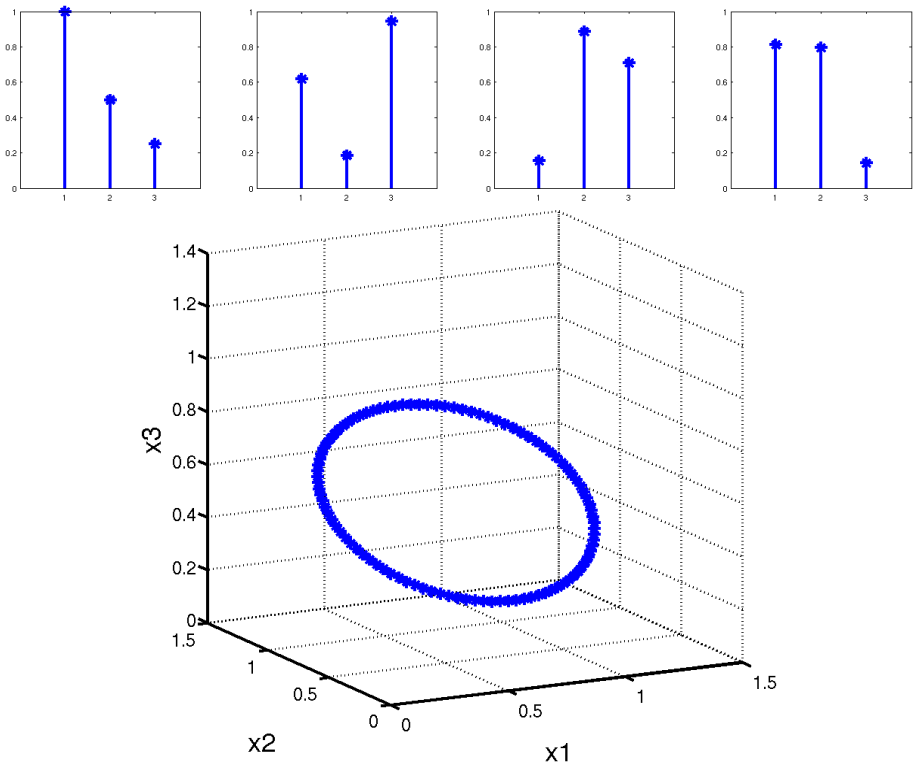

Figure 1.4: The space formed by a three-pixel template translated with circular boundary condition. The top four figures shows four instances of the signal, each translated by a different amount. The bottom figure shows the one-dimensional manifold embedded in a three-dimensional pixel space formed by the ensemble of the signals, with each marker representing a particular instance.

The interesting aspect of this signal family is that while the signals are three dimensional, PCA is able to capture all the data variances with two dimensions. This can be seen visually from figure 1.4 that the data forms a two-dimensional circle in a three-dimensional

space. However, this family of signals is created with only one degree of freedom, namely the amount of shift of the original template. This suggests that there should be a more compact way to represent this family of signals. More importantly, the generator of the signals is the translation operator, and neither PCA nor ICA / sparse coding is able to discover this cause.

This suggests that one way to build a better model of the sensory inputs is to exploit the transformational relationship within the data. Ignoring these relationships makes understanding very difficult. Previous work such as Locally Linear Embedding (LLE) [Roweis and Saul, 2000] attempts to do so by stitching together local neighborhoods in a piece-wise fashion. While such techniques are great for data visualization, they do not attempt to tease apart the causes of the manifold, nor can they generate new data consistent with the manifold structure. This motivated us to develop a general algorithm that is capable of discovering a compact description of the data manifold by modeling transformation between data points. Such a model will allow us to tease apart the causes of data such as that in figure 1.4.

1.2 Modeling Transformations in Image Sequences

The images which fall upon the retinae change over time due to self motion, object motion and object interactions. While the input is highly dynamic and ever changing, the brain needs to parse and understand the causes of the data to give us a stable percept of the world. The same problem is also faced by machine vision systems.

Although the transformation model we develop in this thesis is generic and can be applied to any signal domain, we concentrate our effort on modeling time-varying images. Below, we give a brief overview of the currently available methods.

1.2.1 Explicit Representation of Transformations

Transformations in natural scenes are a result of movement and dynamical interaction of physical objects in the world. For example, object movements parallel to the visual field produce translation in the visual scene, while perpendicular movements produce scaling. Therefore, one obvious attempt is to model the dynamics of time-varying images with explicit transformations such as translation, rotation and scaling. However, parameterizing such a model turns out to be very difficult, and it is hard to infer the dynamics in the visual scene (computing the amount of transformation needed to produce the image at time $t+1$ from time t).

By restricting the transformations to translation, the transformation parameters can be computed through an exhaustive search scheme. On digitized videos, such exhaustive search can be implemented quickly through convolution. For scaling, exhaustive search can be performed through a spatial pyramid, where each level of the image pyramid represents a different scale of the scene. Such transformation algorithms provide the basis for the state-of-the-art video compression [Wiegand *et al.*, 2003], object tracking, depth extraction from stereo/multi-view camera and video interpolation [Mahajan *et al.*, 2009].

Instead of defining transformations in the image domain, it is also possible to define transformation in a feature domain. Taking a set of features that are used to represent static images, for example, the Gabor filterbank, one can transform each element in the filterbank with explicitly defined transformations, e.g. rotation, scaling, etc. One can treat the set of transformed features as one feature bank and perform image decomposition using existing techniques [Escoda *et al.*, 2009]. By comparing features in the neighboring time points, one can infer the transformation parameters. Leveraging recent image decomposition algorithms [Mallat and Zhang, 1993], [Mairal *et al.*, 2010], this method is often more efficient than exhaustive search algorithms defined in the pixel domain. More importantly, by defining transformation in the feature domain, one is able to parametrize the transformation and modern optimization algorithms can be used to recover the transformation parameters [Kokiopoulou and Frossard, 2009]. The authors showed that when the two images are compared under tangent distance (distance measured perpendicular to the tangent of the manifold), the optimal transformation parameters of translation, rotation and isotropic scaling can be recovered.

1.2.2 Spatio-Temporal Representation of Transformations

Alternatively, as Fahle and Poggio [Fahle and Poggio, 1981] and Adelson and Bergen [Adelson and Bergen, 1985] have pointed out, image motion is characterized by orientation in space-time. One may therefore design filters in space-time to capture such dependencies [Watson and Albert J. Ahumada, 1985]. Watson and Ahumada used hand-coded separable space-time wavelets and Van Hateren and Ruderman [Hateren and Ruderman, 1998] showed they were able to learn space-time filters that optimally tile the space of blocks of natural image sequence. Olshausen [Olshausen *et al.*, 2002] further improved the model by adding shift invariance to the learning algorithm. The author showed that the learned filters possess similar properties to the space-time receptive fields found in the mammalian early visual cortex.

Each unit in the model is bounded to a specific spatial location and orientation in the spatial-temporal plane. Therefore, the unit provides a description of the local velocity content instead of visual speed of the object. This is commonly known as the aperture problem [Hildreth, 1984] and further processing is required to turn the unit activations to object

transformations.

1.2.3 Hierarchical Representation of Transformations

Neurophysiological data suggest that processing in the brain is organized in a hierarchical fashion, in that more abstract information, e.g. the identity of an object, is represented further in the processing stream than lower level information, such as edges. Various attempts have been made to construct multi-layer networks to model static images [Hinton and Salakhutdinov, 2006], [Serre *et al.*, 2007]. This concept can also be used to model transformations in time-varying images.

One of the earliest work of this kind is by Simoncelli and Heeger [Simoncelli and Heeger, 1998]. Termed energy model of motion, a hierarchical model is constructed to overcome the aperture problem encountered by typical spatio-temporal filter models, as described in the previous section. Specifically, a spatio-temporal filter bank consisting of quadrature filter pairs is used to filter the image content. The activation of each of the quadrature filter pair is squared and summed to obtain the motion energy. The spatial distribution of the motion energy is extracted using another layer of filters to produce object motion.

Instead of modeling the amplitude distribution of the quadrature pairs, Cadieu and Olshausen [Cadieu and Olshausen, 2009] built a generative model to model the phase component of the first layer basis functions. They showed that the model is able to capture translation, rotation, scaling and local deformation when trained on natural video. While this model is able to reproduce key aspects of the observed response in early visual cortex, one drawback is that a particular unit in the top layer is only able to produce a small change in the image domain. Therefore, multiple units have to be active when for example a bar moves across the scene and requires further processing to convert unit activity into transformation.

1.2.4 Bilinear and Lie Group Representation of Transformations

Another way to representing transformation in time-varying images is with Lie group operators. In particular one assumes that the infinitesimal change of a movie x at time s can be modeled by a linear operator A

$$\frac{\partial x\left(s\right)}{\partial s} = A\ x\left(s\right) \tag{1.1}$$

where $s \in \Re$, $x\left(s\right) \in \Re^N$ is a N dimensional signal evaluated at s, $A \in \Re^{N \times N}$ is an infinitesimal transformation operator and the generator of the Lie group. The solution of

the above differential equation is

$$x(s) = e^{As}x(0) = T(s)\,x(0) \tag{1.2}$$

e^{As} is a matrix exponential defined by the Taylor expansion and $x(s)$ can be obtained by transforming the original template $x(0)$ with the operator A by an amount s.

The advantage of this model is that it can facilitate transformation with arbitrarily large range and has been proposed for application in image processing [Nordberg, 1994] and computer vision [Van Gool *et al.*, 1995]. While these work focuses on the inference problem with fixed transformations, learning the transformation in this framework has been proposed by [Rao and Ruderman, 1999], [Olshausen *et al.*, 2007], and [Miao and Rao, 2007]. Specifically, one aim to find the optimal operator A that satisfy the following objective functions

$$\underset{A}{\operatorname{argmin}} \sum_{t} \left|\left| x^{(t+1)} - e^{As^{(t)}} x^{(t)} \right|\right|_2^2 \tag{1.3}$$

where t is the t^{th} sample in the training set. In practice, equation 1.3 is very difficult to optimize because

- learning is computationally expensive due to the matrix exponential
- the inference is full of local minima

Therefore, to make learning intractable, the authors propose to approximate the full model with its first order Taylor expansion. Equation 1.2 is then approximated by:

$$\begin{aligned} x(s) &\sim x(0) + sAx(0) \\ x(s) - x(0) &= \Delta x = sAx(0) \end{aligned} \tag{1.4}$$

Because it is a linear approximation, inference and learning with this simplified model is easy. However, by comparing against equation 1.1, it is easy to see that this linear model assumes that the two image frames contain only infinitesimal changes. This is generally not true for videos that are sampled at 25 frames per second. As a result, the learned operators converges to the same transformation and each one is selective for a different speed. Alternatively, [Miao and Rao, 2007] used toy data with only small transformation in each image pairs.

1.3 Thesis Outline

In chapter 2, I describe a continuous transformation model formulated with Lie Groups. In contrast to previous attempts where the inference and learning are done with a linearized

model, I propose methods that allow us to work with the full model described in equation 1.2. In particular, I propose

- an efficient and tractable learning rule using the full model
- an inference objective that avoids local minima

In chapter 3, I demonstrate the proposed inference algorithm is able to correctly recover the transformation coefficients with high accuracy when given images transformed with pre-determined operators. I show that the learning algorithm is able to recover the transformation operators when trained on pairs of images contain pre-determined transformations. When trained on pairs of natural movie patches, the learning algorithm discovers operators performing transformations such as translation, intensity scaling, contrast enhancement and local deformations. In addition, I show that the learned models provide a better description of the underlying transformation both qualitatively and quantitatively compare to commonly used motion models.

In chapter 4, I show that the proposed transformation model can be used for video coding. In current video coding standards, the frame-to-frame motion is estimated and compensated with an exhaustive search motion model. I propose to replace this motion model with the continuous motion model learned on natural videos. I show that doing so leads to a coding gain in the low and high bit regime.

Chapter 2

Learning Continuous Transformation: Model and Implementation

2.1 Continuous Transformation Model

To remind the readers, the continuous transformation model is formated as

$$Y = e^{As}X + \eta \tag{2.1}$$

where $X, Y \in \Re^{N \times M}$ contains M pairs of signals of dimension N. Notice that X and Y do not need to be related in time but can be related arbitrarily and is described by the matrix A. η represents Gaussian I.I.D. noise. The two major hurdles that prevent one from learning transformation with the full exponential model are

- **Computational Expensive Learning Stage**

  Matrix exponential are computational intensive operations. It is on the order of $O(N^3)$ for the evaluation of e^{As}. Moreover, it is necessary to compute $\frac{dA}{de^A}$ when optimizing A. This derivative has computational complexity $O(N^6)$ [Ortiz *et al.*, 2001] when computed naively (the derivative $\frac{da_{ij}}{d(e^A)_{kl}}$ is $O(N^2)$ and there are four set of indexes [i,j,k,l]). This makes learning intractable for large N.

- **Non-convex Inference**

  The inference procedure, which is the search for optimal coefficient s under equation 1.3 when the transformation matrix A is fixed, is unfortunately highly non-convex. This makes gradient descent based optimization algorithms very difficult as the inference landscape is plagued with many local minima and can be easily trapped in a local minimum. To illustrate, a 11×11 white noise image patch is generated and it is translated 3 pixels to the left to create another image. The error function (equation 1.3) is plotted

as a function of the transformation coefficient s, shown as the red solid line in figure 2.1.

Figure 2.1: This figure shows the error function described in equation 1.3 for a pair of randomly generated white noise patch, one shifted 3 pixels to the left from another. There are many local minima in the objective function.

[Miao and Rao, 2007] proposed to use random initial condition with multiple repeats and choose the solution with lowest energy. Not only it is computationally expensive, the number of repeats required grows exponentially with the number of transformations in the model. This makes this approach only valid for small number of transformations.

The issues listed above strongly limited the application of the Lie group transformation framework in practice. For the rest of this chapter, we will provide solution to all these problems.

2.1.1 Eigen Decomposition

To make learning feasible, we exploited that the matrix A can be re-parametrized as

$$A = U\Lambda U^{-1} \tag{2.2}$$

where $U \in \mathbb{C}^{N\times N}$ is an invertible matrix and $\Lambda \in \mathbb{C}^{N\times N}$ is a diagonal matrix. The U matrix in this formulation is not required to be orthonormal, in contrast to PCA or SVD. This setup is also proposed by [Culpepper and Olshausen, 2009] to solve similar problems around the

same time this work is being developed.

The advantage of this representation is that the matrix exponential in equation 1.2 simplifies to

$$e^{U\Lambda U^{-1}s} = I + U\Lambda U^{-1}s + \frac{1}{2}U\Lambda U^{-1}U\Lambda U^{-1}s^2 + \ldots = Ue^{s\Lambda}U^{-1} \tag{2.3}$$

As shown, under this formulation the matrix exponential is only applied to the diagonal matrix $s\Lambda$, which is equivalent to a point-wise exponential on each diagonal element. Hence, we have reduced a full matrix exponential to two matrix multiplications and a point-wise exponential.

2.1.2 Computational Complexity of Learning

The model now becomes

$$Y = Ue^{s\Lambda}U^{-1}X + \eta \tag{2.4}$$

As a reminder, learning in this thesis refers to the optimization of the model under equation 1.3. In particular, the optimization equation is

$$\underset{\Lambda,U}{\operatorname{argmin}} \sum_{m=1}^{M} \left(y^{(m)} - Ue^{s\Lambda}U^{-1}x^{(m)}\right)^2 \tag{2.5}$$

The computational complexity of learning (and inference) of the model is dominated by the evaluation of $e^{sA}X$. The most expensive part of this to compute the matrix exponential, which is $O(N^3)$. The entire operator is then $O(N^3)$. In contrast, by using the representation in equation 2.3, when the vector X is multiplied sequentially from right to left, the only operation involved is a vector-matrix multiplication, which has computational complexity $O(N^2)$. As a result, we have reduced the computational complexity of learning to $O(N^2)$, which is the complexity of a linear model.

2.1.3 Adaptive Inference

Inference in this thesis refers to the search for optimal coefficient s such that $Y = e^{As}X$. To remind the readers, the inference objective is

$$\underset{s}{\operatorname{argmin}} \left|\left|Y - Ue^{s\Lambda}U^{-1}X\right|\right|^2 \tag{2.6}$$

As have mentioned in the previous section, there are multiple local minima in the inference energy landscape and any gradient-based optimization algorithm is likely to be trapped in

a sub-optimal solution.

To overcome the non-convexity of the optimization problem, we introduce an inference approach that adaptively smooths the objective function. This is inspired by the coarse-to-fine matching scheme used to compute optic flow [Lucas and Kanade, 1981], [Black and Jepson, 1996] and image matching [Vasconcelos and Lippman, 1997]. The idea behind this matching scheme is based on the observation that it is easier to find the global minimum under transformations such as rotation, translation and scaling with lower resolution images. The coarse-to-fine algorithm avoids local minima by first match matches at lower resolution and uses the result to initialize the matching at higher resolution.

A similar procedure can be implemented using the Lie group operators. Specifically, the image smoothing is achieved by averaging over all possible range of transformations weighted by a Gaussian distribution

$$T\left(\mu_s, \sigma_s\right) \quad = \quad \int_{-\infty}^{\infty} T\left(s\right) \frac{1}{\sqrt{2\pi}\sigma_s} e^{\frac{||s-\mu_s||^2}{2\sigma_s^2}} ds$$

The above integral can be solved analytically and the result is

$$T(\mu_s, \sigma_s) = U e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2 \sigma_s^2} U^{-1} \tag{2.7}$$

The coarse-to-fine scheme introduce an extra term in the inference objective function and new objective function is

$$\underset{\mu_s, \sigma_s}{\operatorname{argmin}} \left|\left| Y - U e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2 \sigma_s^2} U^{-1} X \right|\right|^2 \tag{2.8}$$

The advantage of this inference scheme can be seen in figure 2.2, where the reconstruction error of equation 2.8 is plotted as a function of μ_s with difference values of σ_s. As shown, the inference landscape becomes smoother as the value of σ_s increases and eventually the objective function becomes convex. The introduction of σ_s variable thus opens up paths to the minimum and helps the optimization procedure from getting into local minima.

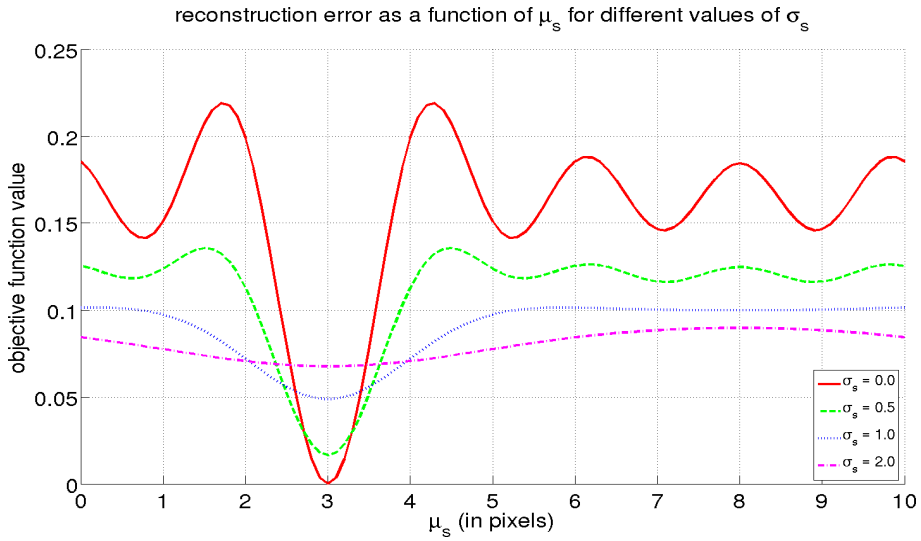

Figure 2.2: This figure shows the reconstruction error as a function of μ_s with different values of σ_s. It shows that the inference landscape only have one minimum for large σ_s values.

The major advantage of this inference scheme over traditional coarse-to-fine matching is that the blurring is part of the objective function. In most coarse-to-fine matching, the lower resolution images are constructed using a multi-resolution pyramid with fixed scales. In our inference method, the blurring is continuous and the optimal level of smoothing is found by the optimization procedure. More importantly, the blurring is operator specific. This makes our inference method valid even when applied to non-imagery data, such as DCT coefficients, where simple spatial blurring makes less sense.

As a final note, the smoothing operator can be viewed as a transformation operator in the form of

$$T(\sigma_s) = Ue^{\frac{1}{2}\Lambda^2\sigma_s^2}U^{-1}$$

2.2 Multiple Transformations

There are often more than one type of transformation present in natural videos. To accommodate multiple transformations in the model, the operators can be concatenated in the following way:

$$T_{multi}\left(s_{\mu}, s_{\sigma}\right) = T_1\left(s_{\mu_1}, s_{\sigma_1}\right) T_2\left(s_{\mu_2}, s_{\sigma_2}\right) ... = \prod_k T_k\left(s_{\mu_k}, s_{\sigma_k}\right) \tag{2.9}$$

$$T_k\left(s_{\mu_k}, s_{\sigma_k}\right) = U_k e^{s_{\mu_k}\Lambda_k} e^{\frac{1}{2}\Lambda_k^2 s_{\sigma_k}^2} U_k^{-1} \tag{2.10}$$

where k indexes the transform. Note that the transformations $T\left(s_{\mu_1}, s_{\sigma_1}\right)$ do not in general commute, and thus the ordering of the terms in the product must be maintained.

Because of the fixed ordering of transformations and the non-commutativity, the multiple transformation case no longer strictly constitutes a Lie group for most choices of transformation generators. Describing the group structure of this new model is a goal of future work. For the present purposes of this thesis we note that transformations observed from video transformations such as affine transformations, brightness scaling and contrast scaling can still be reached using the model in equation 2.9, though choice of coefficient values s_{μ_k} may depend heavily on the order of terms in the product.

2.3 Regularization

2.3.0.1 Regularization on U

There is a degeneracy between the norm of U and U^{-1}, in that if the norm of U grows by a scale factor α, the norm of U^{-1} shrinks by α, and the norm of A stays the same. Although theoretically this does not pose a problem, in practice it makes learning very unstable. To over come this problem, we propose a renormalization step at the end of each iteration such that U and U^{-1} have the same norm. In particular, let us introduce a diagonal matrix R. We notice that the following equation would hold given any non-zero diagonal elements

$$\begin{aligned} A &= V\Lambda V^{-1} \\ &= VRR^{-1}\Lambda V^{-1} \\ &= VR\Lambda R^{-1}V^{-1} \\ &= (VR)\Lambda(VR)^{-1} \end{aligned}$$

by setting

$$U = VR$$

we have

$$A = U\Lambda U^{-1}$$

where R represent the degeneracy in the set of U we are allowed in our decomposition. We

hence propose to regularize U by choosing R such that the joint power is minimized

$$R = \underset{R}{\operatorname{argmin}} \sum_i \sum_j V_{ij}^2 R_{jj}^2 + \sum_i \sum_j (R^{-1})_{jj}^2 (V^{-1})_{ji}^2 \tag{2.11}$$

and its analytical solution is

$$R_{jj} = \left[\frac{\sum_i (V^{-1})_{ji}^2}{\sum_i V_{ij}^2} \right]^{\frac{1}{2}} \tag{2.12}$$

2.3.0.2 Regularization on s_σ

A small L2 penalty is placed on s_σ to encourage it to shrink to zero.

2.3.0.3 Regularization via Manifold Distance

In order to encourage the operators to transform between image patches in the most direct way possible, a penalty is placed on the total distance traveled by the operator between the two patches. The total distance can be computed by summing over all the infinitesimal changes that leads from the starting patch, denoted $x(0)$, to the ending patch, denoted $x(s_\mu)$. Specifically, the distance measure for a particular transform k is

$$d\left(T_k\left(\mu_{sk}, \sigma_{sk}\right); x(0)\right) = \int_{\tau=0}^{s_k} ||\dot{x}_k(\tau)||_2 d\tau \tag{2.13}$$

Expanding the equation we have

$$\begin{aligned} d\left(T_k\left(\mu_{sk}, \sigma_{sk}\right); x_k(0)\right) &= \int_{\tau=0}^{s_k} ||A_k x_k(\tau)||_2 d\tau & (2.14) \\ &= \int_{\tau=0}^{s_k} ||A_k e^{A_k \tau} x_k(0)||_2 d\tau & (2.15) \end{aligned}$$

Finding a closed form solution for the above integral is difficult, but it can be approximated well using a linearization around $\tau = \frac{s}{2}$,

$$d\left(T_k\left(\mu_{sk}, \sigma_{sk}\right); x_k(0)\right) \approx \mu_{sk} ||A_k e^{A_k \frac{\mu_{sk}}{2}} x_k(0)||_2 \tag{2.16}$$

Numerical integration of equation 2.15 is compared against this approximation, shown in figure 2.3. We can conclude from the figure that the zeroth order approximation closely resembles its true value, especially at small s_μ values, where most of the coefficients lie in practice.

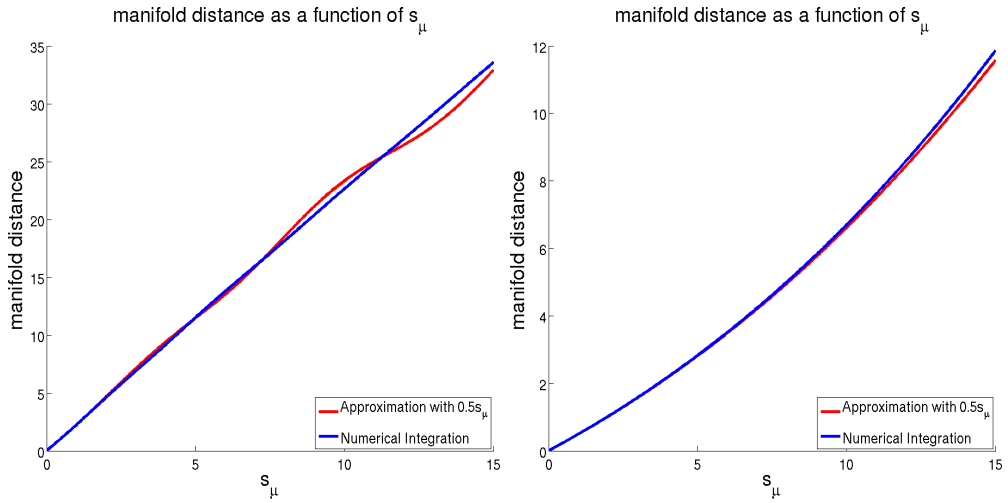

Figure 2.3: This figure shows two examples of comparison between the zeroth order approximation (red) to the manifold distance vs its numerical integration (blue). The transformation operators used here are picked from a subset of operators learned from natural movies.

$$d\left(T_{\text{multi}}\left(\mu_s, \sigma_s\right); x(0)\right) = \sum_k d\left(T_k\left(\mu_{sk}, \sigma_{sk}\right); x_k(0)\right) \tag{2.17}$$

Since this penalty is applied individually to each operator, it also acts similarly to an $L1$ penalty on the path length of the transformations. This penalty will encourage travel between 2 points to occur via a path described by a single transformation, rather than by a longer path described by multiple transformations. In other words, this penalty will encourage independence amongst learned operators.

2.3.1 The Complete Model

Given the above conditions, the full model's objective function is thus

$$\begin{aligned} E\left(\mu_s, \sigma_s, U, \Lambda, x\right) = \quad & \frac{1}{2\widetilde{\sigma}^2} \sum_m \left|\left|y^{(m)} - T_{multi}\left(\mu_s^{(m)}, \sigma_s^{(m)}\right) x^{(m)}\right|\right|_2^2 \\ & + \eta_m \sum_m \sum_k \mu_{sk}^{(m)} ||A_k e^{\frac{\mu_{sk}^{(m)}}{2} A_k} x_{mk}||_2 \\ & + \eta_\sigma \sum_m \sum_k (\sigma_{sk}^{(m)})^2 \end{aligned} \tag{2.18}$$

2.4 Inference and Learning with Region Buffer

When considering pairs of natural movies patches at a fixed spatial location, it is not uncommon to find that the underlying object travels outside of the patch. This situation is graphically illustrated in figure 2.4.

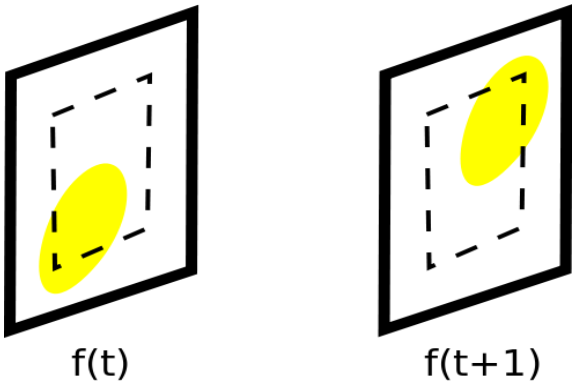

Figure 2.4: An illustration of the motivation for the use of region buffer during inference and learning. The dotted line represents the boundary of the buffer region.

Therefore, to account for such scenarios, we introduce an image buffer to the inference and learning algorithm. In particular, the image buffer is an extended region around the original image patch. The inference and learning algorithm 'sees' both image patch as well as the buffer region. However, during inference and learning, the reconstruction error within the buffer region does not count towards the objective function. This setup therefore allows the algorithm to bring information inside the buffer region to reconstruction the central part of

the image patch, while without getting penalized for not reconstructing details inside the buffer region.

2.5 A Probabilistic Model

This model can be recasted in a probabilistic framework as follows:

$$\begin{aligned} p(x, s, U, \Lambda) &= \frac{1}{Z} \exp(E_{full}) \\ p(x|s, U, \Lambda) &\propto \exp\left(\eta_x ||Y - T_{multi}(s_\mu, s_\sigma) X||^2\right) \\ p(s_\mu) &\propto \exp\left(-\eta_{s_\mu} \sum_k |s_{\mu_k}|\right) \end{aligned}$$

where Z is the partition function.

2.6 Model Estimation

In equation 2.5, the coefficient values s are latent variables, hence unknown. This poses a chicken and egg problem; the model cannot be updated without knowing each individual coefficient but the model is unknown to begin with. In this thesis, we employ a variational Expectation-Maximization (EM) optimization approach [Olshausen and Field, 1997] by alternating between the following two steps:

1. coefficients are inferred for each pair of signal using equation 2.8 with the model fixed.
2. the model is updated with the inferred coefficient values using equation 2.5

For each of the steps, the optimization is carried out using Minfunc [Schmidt, 2009], a gradient Matlab optimization package developed by Mark Schmidt. Please refer to Appendix A for the detail computation of the objective gradients.

2.7 Conclusion and Future Works

In this chapter we have proposed an algorithm for learning continuous transformation models from data. Different from previous attempts, we have introduced a tractable learning algorithm via a reparameterization of the model and an inference algorithm that avoids local minima via adaptive smoothing. Both of these features are key to make the algorithm work

in practice.

There are several lines of possible extensions on the model we proposed here. We list a few of them below:

- The learning algorithm used in this thesis is based on an alternative EM approach, where we replace the expectation over the hidden variable s_μ and s_σ with a single value at the maximum a-posterior (MAP). This type of approximation is valid when the underlying hidden variable distribution is unimodal and is heavily peaked. For the model proposed here, we have relatively little intuition of how the underlying hidden variable distribution should behave and thus the solution maybe biased by this learning method.

  The reason for using the MAP solution to approximate the expectation is because finding the expectation is very slow and often intractable. However, with the recent advances in Monte-Carlo sampling and the use of highly parallel computing architectures such as GPU, various groups have shown the it is now plausible to estimate the true posterior. Therefore, a interesting line of future work maybe to develop a sampler based learning algorithm and observe how the learned solutions change.

- The continuous transformation model can be used to understand the representation of natural scenes. Olshausen and Field [Olshausen and Field, 1996] showed that local image structures can be captured by using an overcomplete linear generative model, known as sparse coding. However, the dictionary elements show a large degree of redundancy, in that multiple elements have very similar spatial structure but differ in spatial positions and orientation. We believe that the sparse coding model can be greatly enhanced if the transformation is represented explicitly using the continuous transformation model. In particular, we propose

$$x = \sum_i \beta_i T(s_\mu, s_\sigma)\Phi_i + \sum_i |\beta_i| \tag{2.19}$$

  where x is an image patch, Φ is the overcomplete dictionary and β are the corresponding coefficients. In this model, there is a common set of transformations shared between all the basis functions and each dictionary element Φ_i is independently transformed by $T(s_\mu, s_\sigma)$ and summed to form the image patch x. We propose to learn both the transformation T and the basis functions Φ.

- There are two main attributes in sensory data, namely 'what' and 'where'. For example, in visual data, the 'what' attribute can describe the identity of the objects in the image and the 'where' attribute can describe the physical location of the objects.

The transformation model developed in this chapter describes the relationship between observed variables, hence the 'where' attribute. PCA, ICA and sparse coding are often used to recover the 'what' attribute.

One way to simultaneously recover both 'what' and 'where' is to construct a dynamical model with hidden variables. Specifically, the observed variables are generated with a set of features and the dynamics of the features are modeled . One of the most common dynamical model with hidden variables is a hidden Markov model (HMM), shown in figure 2.5. In a hidden Markov model, the dynamics of the hidden variable x is modeled with a linear operator A, to generate the observed variable y.

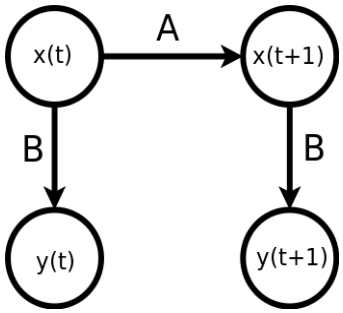

Figure 2.5: The graphical model of a hidden Markov model.

In the standard HMM depicted above, the dynamics of the hidden variable is modeled with the first order Taylor approximation of the full exponential model described in equation 1.2. As explained throughout this chapter, we are able to make a better model by learning with the full exponential model using algorithms we have developed in this chapter.

Chapter 3

Learning Continuous Transformation from Video

3.1 Overview

This chapter is divided into two parts. The first part of this chapter concentrates on demonstrating the correctness of the inference and learning algorithm for the continuous transformation model developed in chapter 2. We show that when pairs of images are generated with previously determined transformations, the inference algorithm can recover the transform parameters with high accuracy. In addition, the transformation operators themselves can be recovered by the learning algorithm when trained on image pairs with pre-determined transformations. We will compare our results with findings from previous work whenever possible. There are only a handful of groups that have attempted to solve similar problems and we believe our work is the first that tries to quantitatively analyze such model.

In the second part of this chapter, we show the transformations learned on natural time-varying images. Here the actual underlying transform operators are unknown. Some of the learned operators exhibit global behavior, such as intensity scaling, local contrast enhancement or translation; other transformations only operate on a part of the patch. Some of the learned transformations are very hard to interpret and methods are still need to be developed to understand the learned operators. Finally, we show quantitative results on how well the learned models describe the image when as a function of degree of freedom in the model and we compare against models that are currently used for motion modeling.

3.2 Parameter Recovery with Affine Transformed Images

The proposed inference algorithm is tested with pairs of images that are generated with affine transformation. The algorithm is then used to infer the transform coefficients given the pairs of images and the affine operators. The inferred coefficients and the true coefficients are thus compared and reported.

3.2.1 Affine Transformed Dataset

Affine transformation is a family of transformation that can be described by a linear transformation in the vector space followed by a translation:

$$x' = Ax + b \tag{3.1}$$

where x and x' are typically pixel coordinates when the transform is applied to images. There are seven types of image transformation that can be described through affine transformation:

- horizontal and vertical translation
- rotation
- horizontal and vertical scaling
- horizontal and vertical sheer

Affine transformation serves as a perfect candidate to test the proposed inference / learning framework because they can be easily parametrized. To generate test data using affine transforms, 13×13 image patches are randomly selected and cropped out from a corpus of video clips obtained from the BBC documentary series *Animal World*. The videos can be obtained from Hans Van Hateren's repository at *http://hlab.phys.rug.nl/vidlib*. The videos contain footage of animals dwelling in their natural habitat and they consist of various types of motion such as camera motion, object motion, tracking and occlusion. Some example frames are shown in figure 3.1.

Each image patch is then independently transformed by **each** of the seven affine transformations listed above, with the degree of transformation chosen randomly from a uniform distribution with range specified in the table below.

Transformation	Range	comments
Translation	[-0.5, 0.5]	0.5 corresponds to shifting half the image.
Rotation	[-180°, 180°]	
Scaling	[0.7, 1.3]	scaling between 70% to 130%
Skew	[0.7, 1.3]	skew between 70% to 130%

Figure 3.1: Sample frames exacted from *Animal World*.

Please refer to Appendix B for a detailed treatment of affine transformation generation.

In this experiment, translation is periodic with wrap around boundary condition (e.g. for horizontal translation, image content shifted to the right will wrap around and appear on the left). Rotation and translation are applied at their maximum range. Scaling and skew contain transformations as large as half of the image patch. Transformation larger than these specific range causes severe image degradation.

This dataset is difficult compare to ones used in previous literature because of the large transformation range. Previous attempts tested their inference algorithm with much smaller transformations, for example, [Miao and Rao, 2007] restricted translation to ± 2 pixels and rotation is restricted to $\pm 15°$.

3.2.2 Parameter Recovery Accuracy

In this test, 2000 image patch pairs containing affine transformation are generated. Both inference with adaptive blurring (equation 2.8) and without (equation 2.6) are used to infer the transformation between a given pair of image patches and their performance is compared.

The operators used in this experiment are hardcoded in the model. In particular, translation operators are created using the Fourier transform (see Appendix B for detail) and the rest are learned individually given pairs of images containing only the specific transformation (the detail is described in section 3.3).

All the s_μ coefficients are initialized at 0 and the s_σ coefficients are initialized at 0.2 to start the inference.

The errors between the true coefficient values and the inferred values are reported as percentages computed as follows:

$$error = \frac{|\theta - \hat{\theta}|}{\theta}$$

where θ are the parameters used to generated the image pairs and $\hat{\theta}$ are the parameters inferred using our algorithm.

Figure 3.2 shows the fraction of the recovered coefficients for which the error is equal or less than 1%. The inference algorithm with adaptive blurring outperformed the inference algorithm without blurring. This is expected as it is shown before that inference without adaptive blurring can be easily trapped in local minima and results in sub-optimal coefficients. This result also shows the difficulty of inferring transformation parameters with traditional approaches and the reason why earlier works have so much trouble working with such models.

While one may expect that the error should be consistent across all transformations, the fact that translation performs better than other types of transformations is probably due to the specific operators we used. In particular, the two translation operators are constructed from theory while the rest are learned because they are less straightforward to construct theoretically. The problem is caused by the fact that the training images the algorithm learned on are generated with coordinate-wise affine transform followed by bilinear interpolation. While the proposed model can handle the particular type of transformation, it cannot simultaneously model the effect of bilinear interpolation with the same operator. This causes a

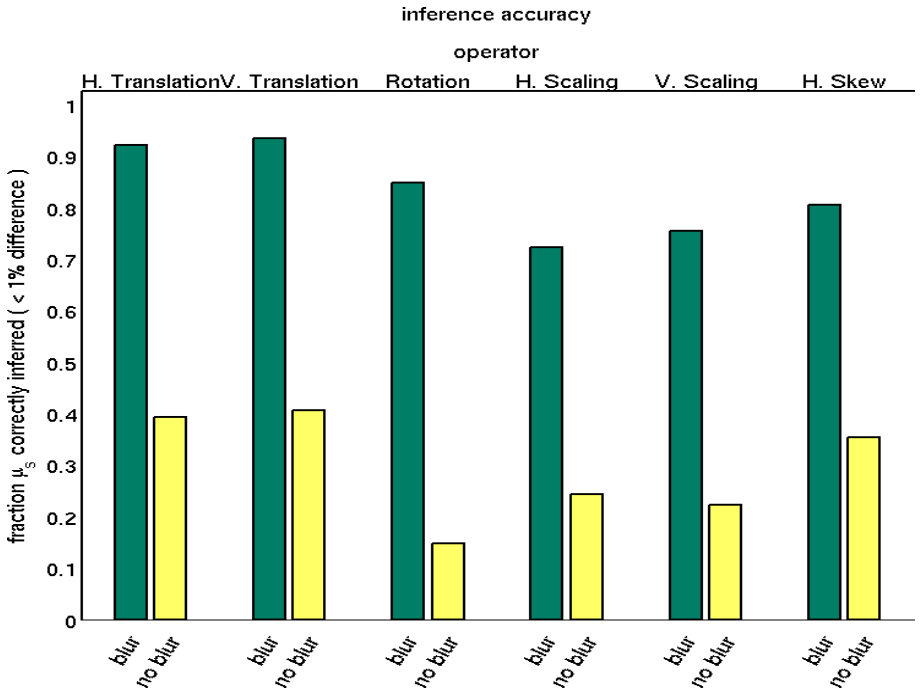

Figure 3.2: The fraction of all simultaneously inferred coefficients which differed by equal or less than 1% from the true coefficient values. Inference with and without adaptive blurring is being compared.

mis-match in the very high frequency components of the image, where bilinear interpolation operates. This mis-match we believe have interfered with the parameter recovery, especially at a high fidelity regime where we require the error to be within 1%.

While the parameter recovery shows a 70% to 90% accuracy, we believe that most of the image patches are indeed reconstructed well, which is the original objective for the inference algorithm. To validate this claim, we compared the error between the reconstructed image and the target image, in terms of peak signal-to-noise ratio (PSNR), defined as follows

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$
$$MSE = ||I - \hat{I}||_2^2$$

where I is the affine transformed image patch and $\hat{I}$ is the reconstructed image patch using our algorithm. MAX_I is the maximum pixel value in the image.

The distribution of PSNR is shown in figure 3.3. As shown, this distribution is bimodal. The small distribution on the left centered at 0dB represents the population of parameters that are wrongly inferred. We believe that larger distribution on the right centered at 35dB contains most of the data whose parameters are correctly inferred. We have verified that for a reconstructed image patch, a PSNR over 30dB appears to be visually similar. The percentage of the reconstruction that exceed the 30dB threshold is 82%.

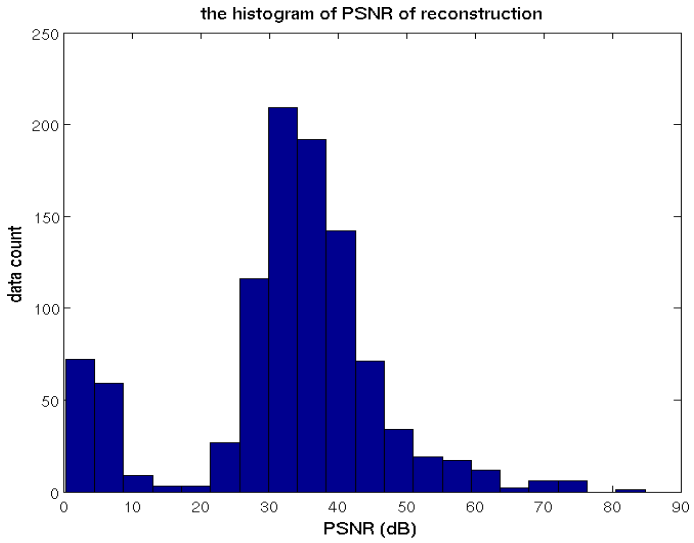

Figure 3.3: The distribution of PSNR for image patches reconstructed using coefficients for all transformations simultaneously inferred with adaptive blurring.

This result confirmed that the proposed inference is working well, even under the circumstances where the testing images contain large range of transformation. We expect that the inference result will be better if one is able to construct 'clean' affine transformation operators. This is under investigation and will be a direction of future work.

3.3 Learning Affine Transformations

In this section, we focus on demonstrating the correctness of the learning algorithm. This is done by showing that our proposed algorithm can recover the operators from images that are transformed with affine transformation. For this experiment, we used the same affine transformed image dataset that is created in the previous section.

3.3.1 Model Initialization

It is important to choose an initialization of the model since the parameter space for learning is large (N free parameters for each Λ and N^2 parameters for each U) and and highly non-convex. For affine transformations, the total power of the image remains constant, and therefore correspond to simply a rotation in the high dimensional image vector space. For the operators in the continuous transformation model to be norm preserving, Λ is restricted to be purely imaginary. Therefore, Λ is initialized to be imaginary only with entries drawn from an uniform distribution between [-0.1, 0.1]. We do not have strong intuition about the initialization of U. Therefore, we simply initialized U to an diagonal matrix with small random Gaussian noise drawn from an uniform distribution between [-0.01, 0.01] added to each entry of the U matrix. The noise provides traction for the learning algorithm to leave from its original initialization.

3.3.2 Convergence Property

The first question we would like to ask is how fast does our learning algorithm converge, if it converges at all. To answer this question, we would like to look at the reconstruction error of the data. In particular, while learning, we use a hold-out set of 200 image pairs that is not used for training. After every iteration, the updated model is applied to this dataset and reconstruction error is recorded. Ideally, the reconstruction error should decrease after each model update and the learning is done when the error plateaus. Figure 3.4 shows the total reconstruction error of the 200 image patches as a function of iteration when learning the vertical and horizontal transformation simultaneously.

Although this is not a direct measure of model convergence, it implies how much the model is changing after each iteration. In the example given above, the learning algorithm converges after approximately 120 iterations with 500 pairs of images used per iteration. The learned model can perfectly reconstruct the data at the end of learning.

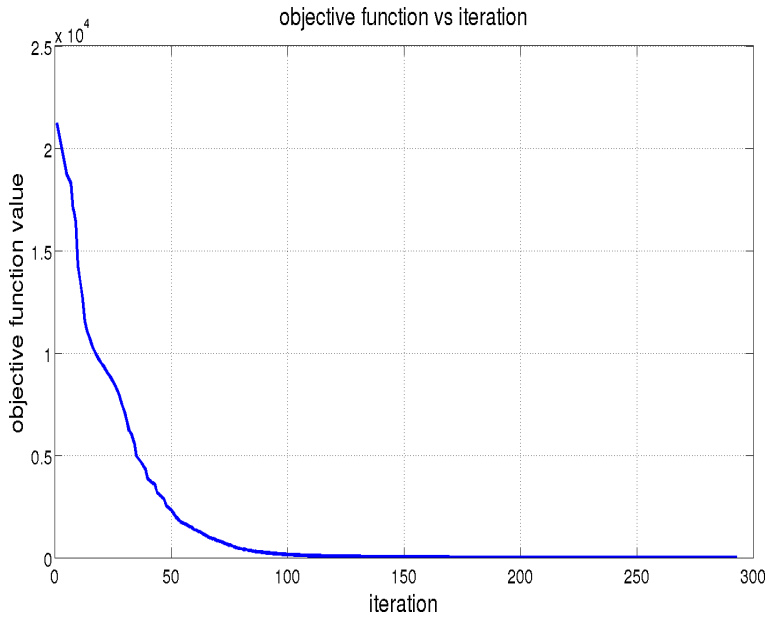

Figure 3.4: Reconstruction error per pixel as learning progresses.

3.3.3 Learned Affine Operators

In this section, the learned affine operators using the proposed learning algorithm are shown and discussed. The horizontal and vertical operators are learned together, while the other transformation operators are learned individually.

The learned translation operators are shown in figure 3.5. In the figure, each square represents one row of the operator matrix, reorganized into the same size as the input image. The position of each square corresponds to the pixel position in the image patch. The raw value within each square corresponds to the weight applied by the operator to the input image. In other words, this figure shows the weighting of the operator at each input pixel location to the output image.

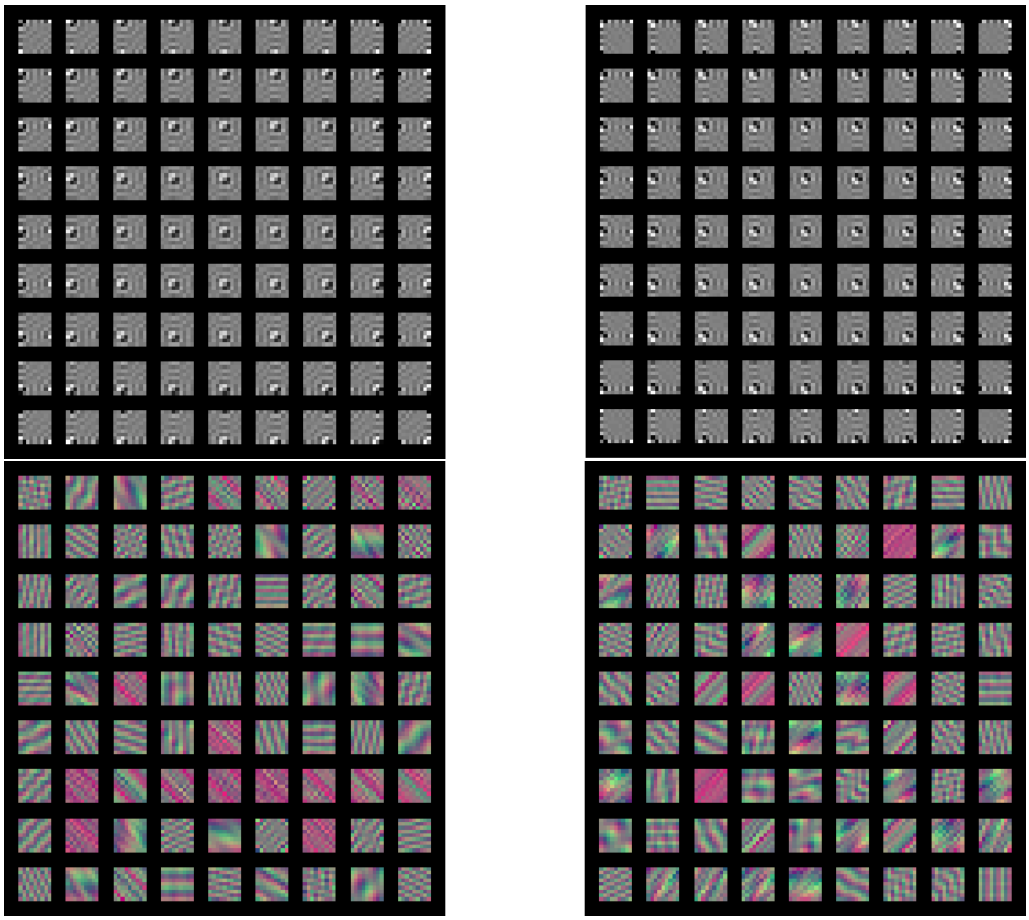

Figure 3.5: Two learned translation operators (top left and top right) and their corresponding Eigen-vectors. Each square represents one row of the operator matrix, reorganized into the same size as the input image. The position of each square corresponds to the pixel position in the image patch. The raw value within each square corresponds to the weight applied by the operator to the input image.The red components denotes real Eigen-vectors and the green denotes imaginary Eigen-vectors.

Recall in the continuous transformation model that the infinitesimal change $\dot{x}$ is represented as a linear combination of the input Ax. Therefore, for image translation, we expect the operator A to be derivative operator oriented in the shift direction. This intuition correlates well with the finding, where the learned translation operators resemble the shape of numerical differentiators. To facilitate translation in the X and Y direction, the two operators need to be directionally orthogonal to each other. This is also evident in the figure.

In addition, the eigen-vectors of the learned operators are sinusoidal and the corresponding

eigen-values are strictly imaginary. This shows that the learned operators is able to recover the original translation operators that is used to generate the training image. The fact that the learned Eigen-vectors do not appear to be the exact Fourier basis is because the phase of the traditional Fourier basis all starts at zero. This is not a requirement to translate an image because it is only the relative phase between sinusoidal components that matters. Therefore, in our model, the learned Eigen-vectors all start at a random phase but its functionality is exactly the same as the Fourier basis.

With the same principle, the rotation operator should correspond to angular derivatives. This indeed can be seen in figure 3.6. The eigen-vectors resembles the Fourier basis on polar coordinates.

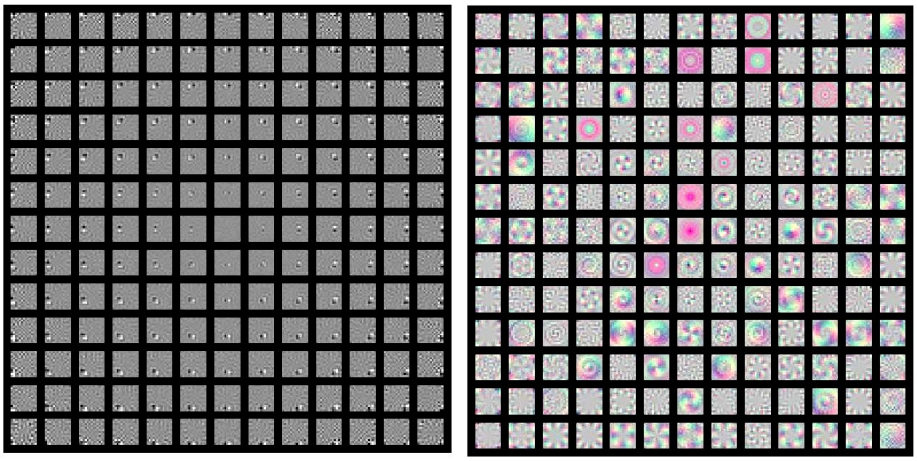

Figure 3.6: The learned rotation operator (left) and its corresponding eigen-vectors(right)

The scaling operators (figure 3.7) resemble derivative operators symmetric about the image center. In addition, the size of the derivative operators increases as they move away from the center, reflecting increasing expansion/contracting away from the center of the image. On the other hand, the skew operators (figure 3.8) are derivative operators symmetric about the diagonal.

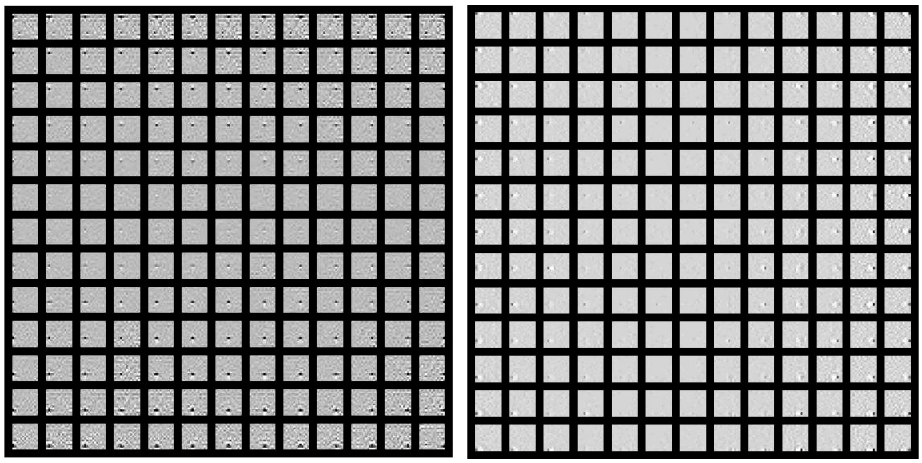

Figure 3.7: The learned scaling operators in the vertical (left) and horizontal direction.

3.3.4 Visualizing Learned Operators

One major advantage of the proposed model has is that image transformation can be easily generated from the model. This can be done by applying different values of s_μ to the model with fixed operators and observe the input image changes under the learned operator. We find this to be a very effective way to visualize the functionality of the learned operators. Figure 3.9 shows a given image patch transformed by various amount under each of the learned affine operators. The image change under different transformation coefficient is also recorded as a movie and can be viewed at https://redwood.berkeley.edu/jwang/affine.html. The movie in fact further validates that our learned algorithm performs correctly.

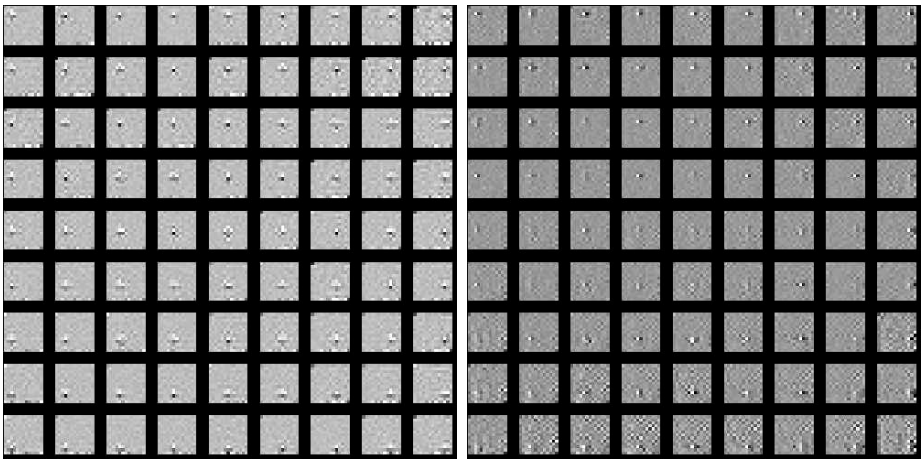

Figure 3.8: The learned skew operators in the vertical (left) and horizontal direction.

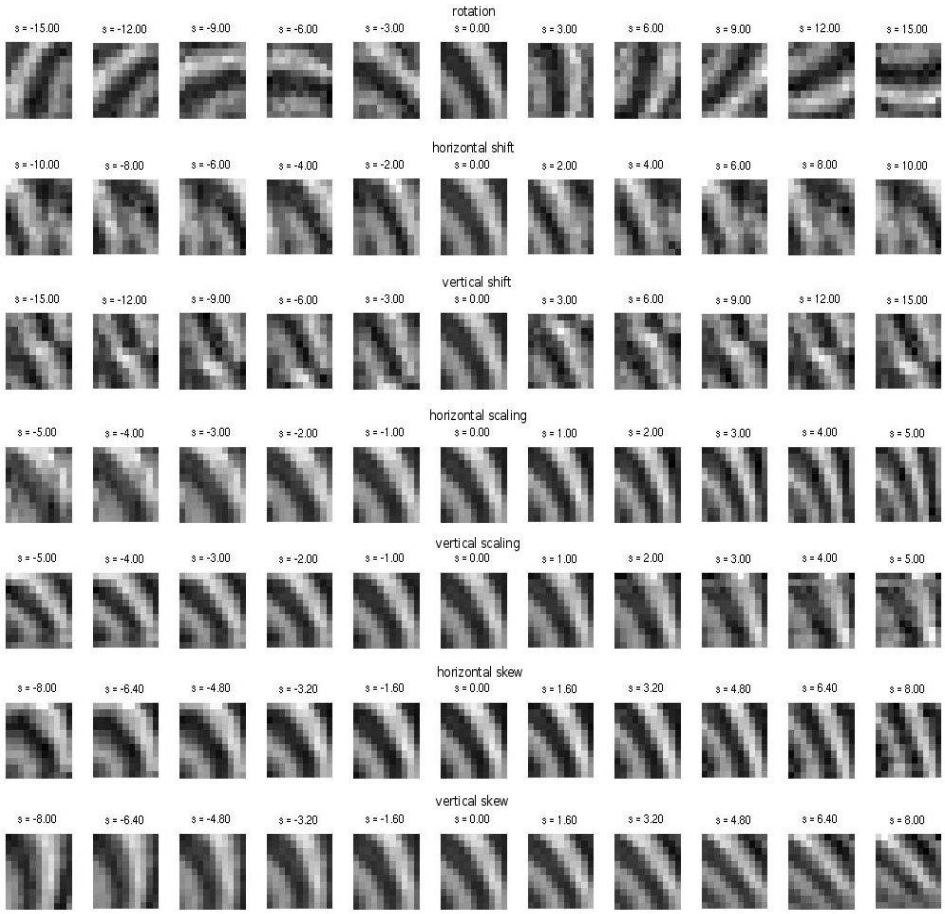

Figure 3.9: Example of the learned affine operators applied with different coefficient value to an image patch.

3.3.5 Linear vs Exponential Model

Finally, we would like to qualitatively compare the difference between the operator learned with the exponential model against its linear approximation. This is done by comparing the difference between images generated with the learned operator and the images generated with an analytical operator with various coefficient values. Figure 3.10 shows such a comparison for a rotation operator. Because we cannot construct the analytical rotation operator, we approximate it by a coordinate-wise rotation operator with bilinear interpolation.

The red line in figure 3.10 is reproduced from figure 7 in [Miao and Rao, 2007]. The operator is learned using a linear transformation model. As we can see, the error between the images generated from the learned and analytical operator grows as a function of rotation angle. This is expected as the linear approximation of the exponential model only holds for small transformation values.

The blue line shows this for the rotation operator learned using the exponential model. In contrast to the linear model, the error stays small over the entire transformation range. The small error is mainly due to the bilinear interpolation. In particular, the model cannot rotate and account for the effect of bilinear interpolation with the same operator. In other words, this small error is caused by the imperfection of the analytical operator, not a weakness of the learning algorithm.

This result further demonstrates the advantage of using a full Lie group model over its first order approximation.

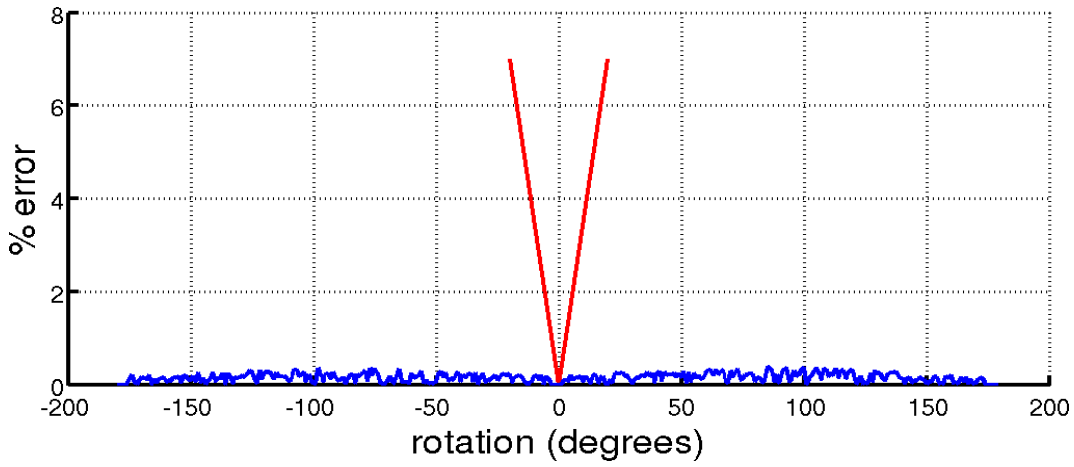

Figure 3.10: This figure compares how much a learned rotation operator deviates from an analytical operator (coordinates-wise rotation operator with bilinear interpolation) by showing the difference between images generated by the operators as a function of rotation angle. The red curve refers to operator learned with first order Taylor expansion (reproduced from Miao and Rao 2007). The blue curve refers to the operator learned with the full model.

3.4 Learning Transformations on Natural Video

One of the goals in this thesis is to learn a set of optimal operators that describes the dynamics between frames in natural video. We attempt to fulfill this goal by learning continuous transformation models on pairs of natural movie patches. The learned models are further quantified by comparing their image reconstruction ability with a rigid translation model, which is often used in video compression and computer vision to describe transition between video frames.

3.4.1 Natural Video Dataset

The training set we used to learn the transformation operators consists of pairs of 17×17 image patches randomly cropped from two neighboring frames in time from a corpus of video clips obtained from *Animal World* (see section 3.2.1 for detail). To ensure that the pair of images contain sufficient motion for the algorithm to learn on, image pairs with a variance smaller than 0.005 on the pixel difference between the pair are rejected.

3.4.2 Learning Procedure

The transformation model in this experiment is trained on 17×17 image patches with a 4 pixel buffer region around each side of the patches. The effective region for which the transformation model learned to reconstruction is therefore the central 9×9 region.

In this experiment, the operators are learned in two different settings. For one setting, the translation operators are pre-coded in the model and the rest of the operators are learned. Specifically, the translation operators appear in the front of the entire operator chain and is applied prior the other learned transformations. The justification for pre-coding translation is two fold:

- firstly, previous work such as [Cadieu and Olshausen, 2009] and [Memisevic and Hinton, 2010] have shown that the vast majority of the transformations found in natural video is translation. In fact [Memisevic and Hinton, 2010] only learned translation operators from natural video.
- secondly, rigid translation operators are widely used in engineering applications such as video coding and we would like to compare our learned results with this motion model. By pre-coding translation in our model allows us to immediately see the gain over the translation only motion model.

In a different setup described next, we learn the entire set of operators all initialized at random.

3.4.3 Learned Operators with Pre-coded Translation

Figure 3.11 and 3.12 show the additional operators learned when three and four operators are learned respectively, with vertical and horizontal translation operators pre-coded.

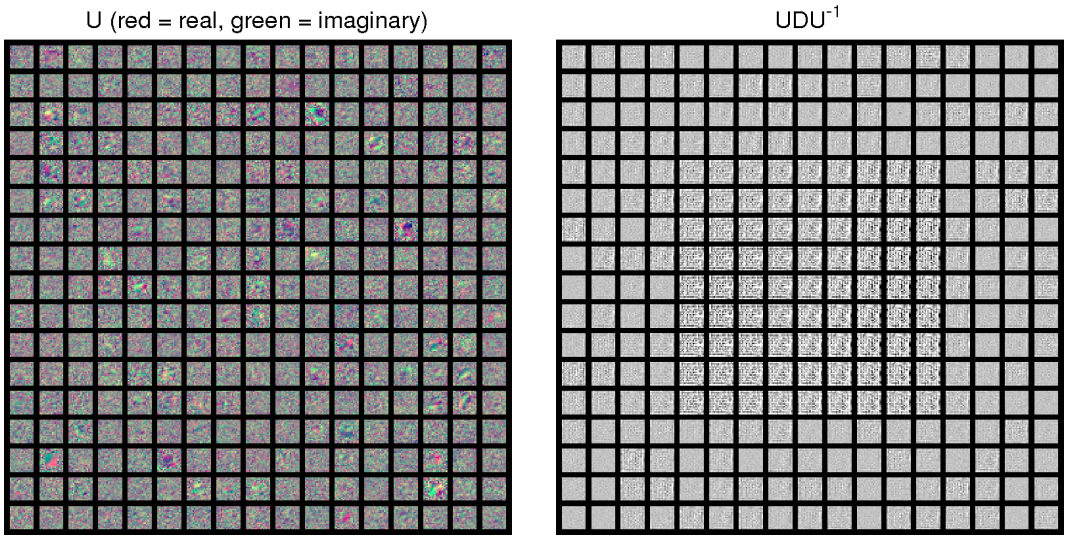

Figure 3.11: This figure shows the Eigen-vectors of the learned operator (left) and the operator matrix (right) when one operator together with two pre-coded translation operators is learned from natural video patches. A 4 pixel buffer region is used during learning.

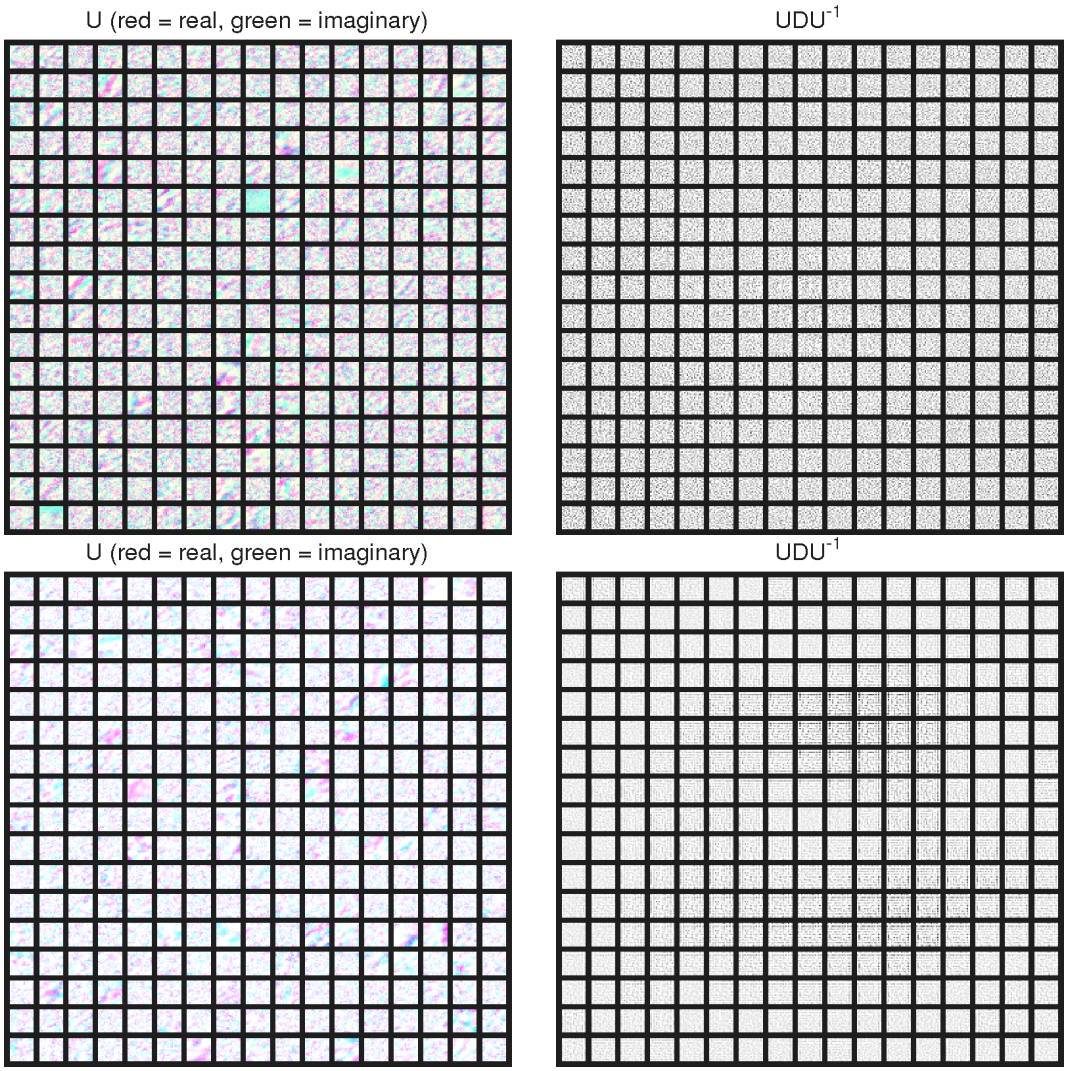

Figure 3.12: This figure shows the Eigen-vectors of the learned operators (left) and the operator matrix (right) when two operators together with two pre-coded translation operators are learned from natural video patches. A 4 pixel buffer region is used during learning.

While one might think that the most interesting aspect of this experiment would be to understand the functionality of the learned operators, we find it extremely difficult to analyze them. The learned operators are very structured and even contain similar structure when we learned three and four operators independently, as shown in figure 3.11 and 3.12. In addition, the learned operators seem to describe changes in lower frequencies, which fits our intuition since most of the power in natural movies lies in the lower frequency range.

Leaving the interpretation of the learned operators aside, we are able to quantify the additional transformation's contribution to the motion model. In particular, 2000 pairs of image patches are cropped from a subset of corpus from *Animal World* which are not used during learning. The reconstruction fidelity in terms of PSNR is compared when we learned one, two, three and four additional operators together with pre-coded translation. The result is shown in figure 3.13.

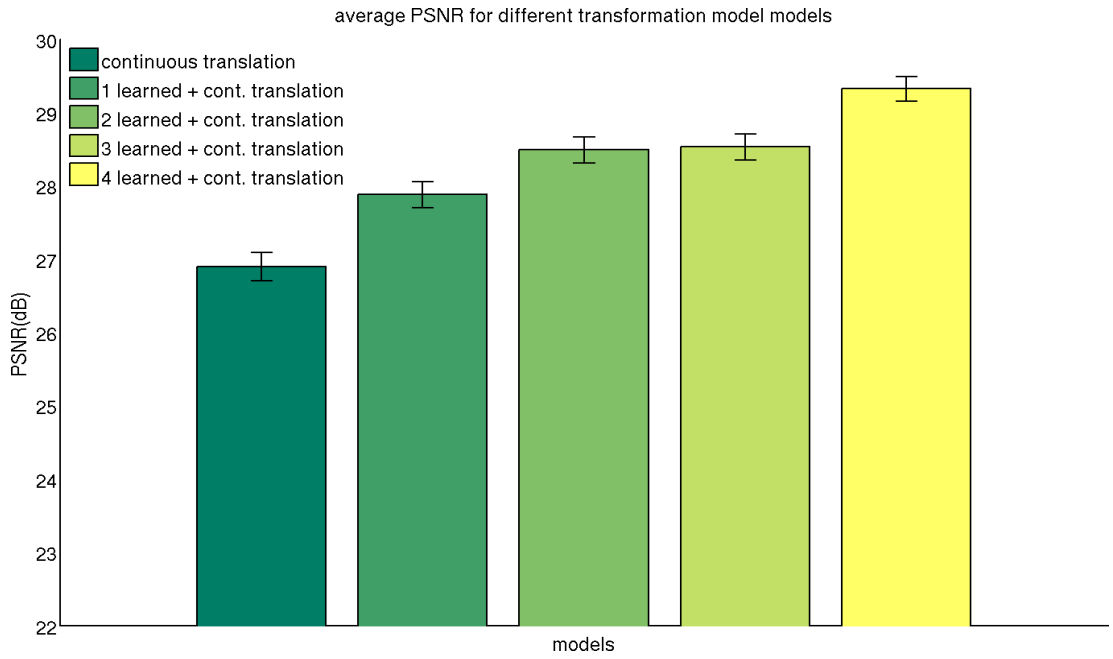

Figure 3.13: The average PSNR of 2000 image patch pairs evaluated using different motion models. The left most bar refers to the result obtained from a translation only model. The rest of the bars correspond to result obtained from 1, 2, 3 and 4 additional operators learned together with vertical and horizontal translation operator pre-coded.

As shown, the PSNR of the motion estimation steady increases when the learned model contains more operators. This is reassuring that each additional transformation encodes something useful and adds to the predictive power of the model.

3.4.4 Learned Operators without Pre-coded Translation

In this experiments, the motion models are learned with all the operators initialized at random. Figure 3.14 and 3.15 shows a subset of the learned transformation from a total of 15 transformations.

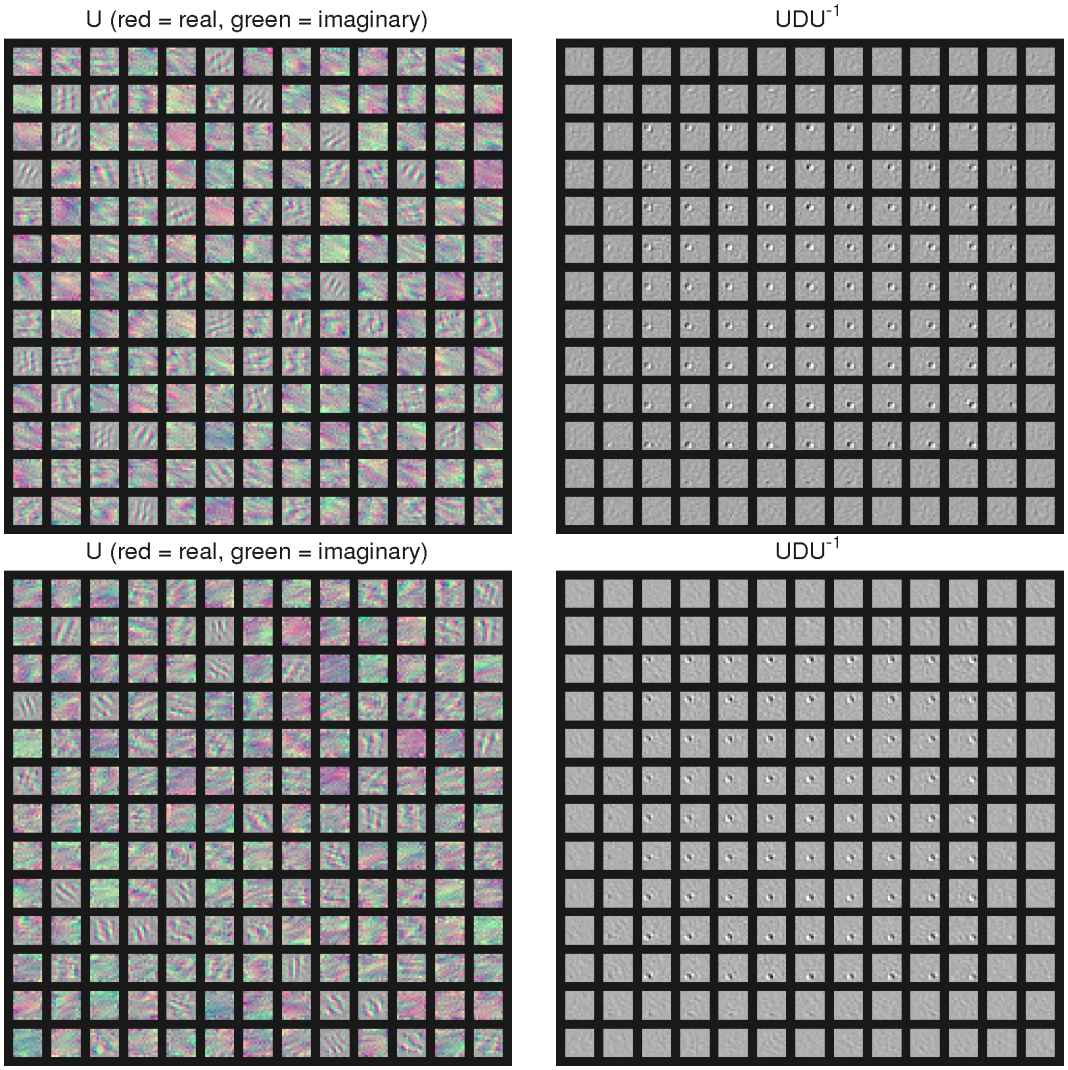

Figure 3.14: Two of the 15 learned operators that behaves like translation operators. The Eigenvectors of the operator is shown on the left and the operator matrix is shown on the right.

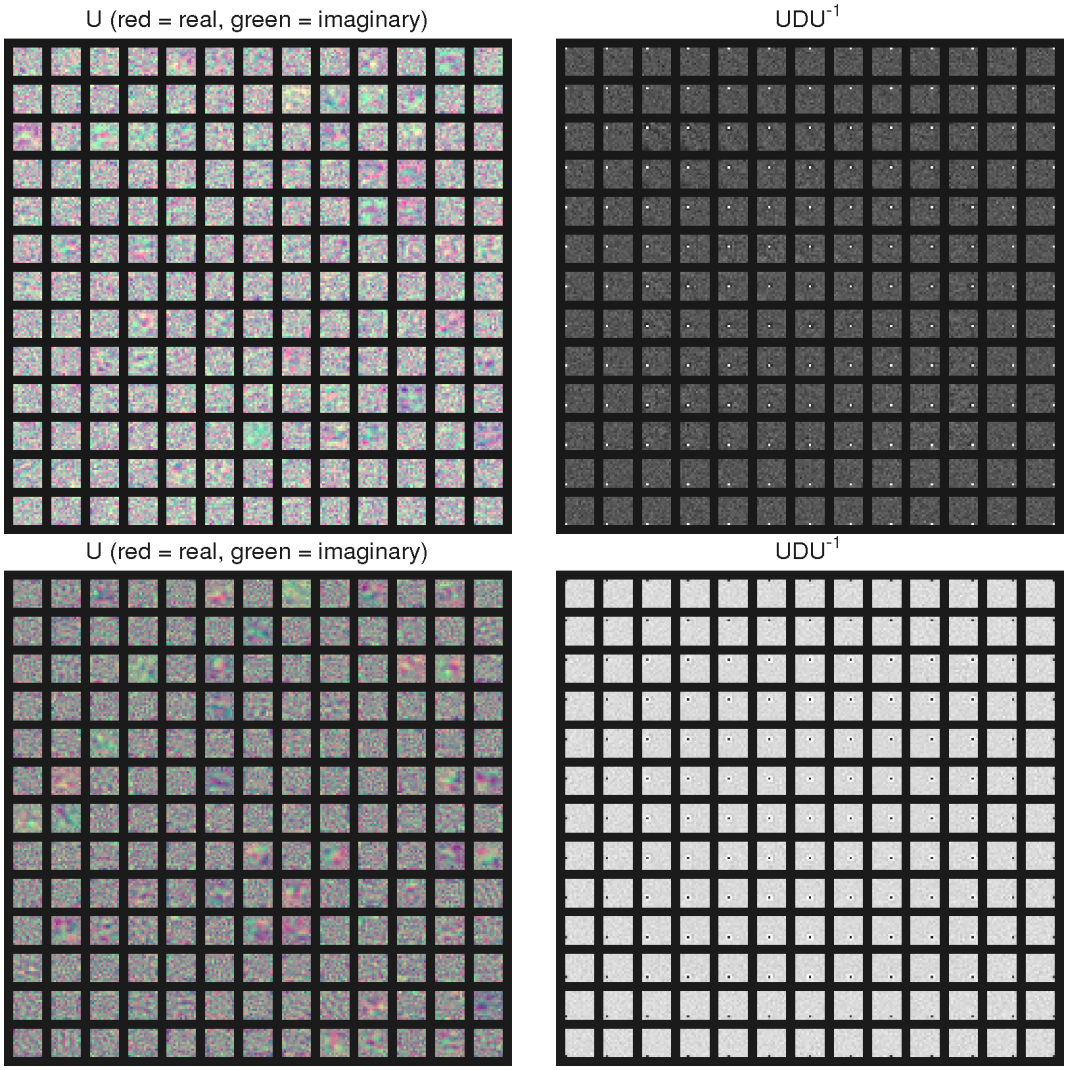

Figure 3.15: Two of 15 learned operators that behaves like intensity adjustment (top) and local contrast adjustment (bottom). The Eigen-vectors of the operator is shown on the left and the operator matrix is shown on the right.

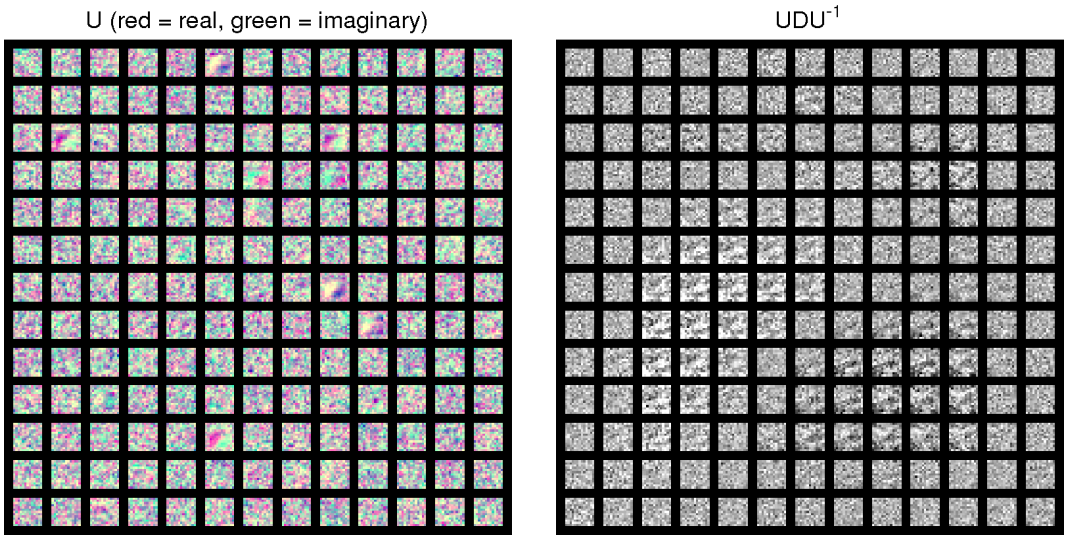

Figure 3.16: An example of the learned operator with unknown functionality when 15 operators are learned from natural video.

In figure 3.14, we see that the learning algorithm learned two translation operators. This confirms our previous experiment setup for which the pre-coded translation operators are useful initializations. Apart from the translation operators, the algorithm also learned global transformations such as intensity scaling and contrast scaling, shown in figure 3.15. Unfortunately, a large portion of the learned transformations are still very difficult to interpret, an example is shown in figure 3.16. On the other hand, this model has a clear advantage in modeling the motion in terms of reconstruction PSNR over all the other models. This is shown figure 3.17

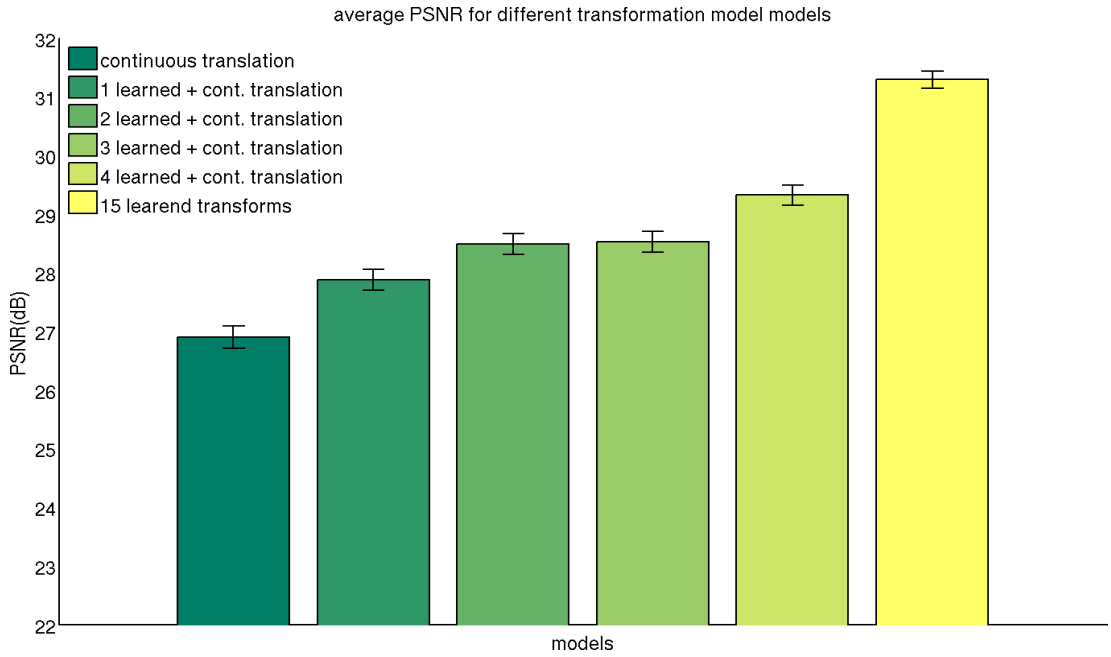

Figure 3.17: The average PSNR of 2000 image patch pairs evaluated using different motion models. The left most bar refers to the result obtained from a translation only model. The rest of the bars correspond to result obtained from 1, 2, 3 and 4 additional operators learned together with vertical and horizontal translation operator pre-coded. The last bar represents result obtained from 15 learned operators.

We have also tried to learn smaller sets of transformation with random initialization, such as 4 or 5 operators. Unfortunately, the learning algorithm does not converge. We believe this is due to the fact that a larger number of transformations are needed to model frame-to-frame changes in patches. When the number of available operators in the model is too few, the learning algorithm is overwhelmed by the diversity of the image transformations and is not able to find a stable solution that satisfy all the training data.

On the other hand, the learning algorithm converges when two translation operators are pre-coded, even when the number of operators available is also small. We hypothesize this is because translation is one of the major type of transformation in natural image patches. This has been confirmed by previous studies, such as [Olshausen *et al.*, 2007], [Cadieu and Olshausen, 2009] and [Memisevic and Hinton, 2010], where a vast majority of transformations the authors learned constitutes translation of different speed and orientation. We further confirmed this as two of the 15 learned transformations constitute translation in orthogonal directions. By pre-coding translation in the model, we therefore made the problem much easier for the algorithm as two operators have already accounted for most of the signal trans-

formation. We have further experimented with learning a smaller set of transformations with two of the operators initialized to translation, but they are not hardcoded, in that they can change as the learning progresses. This also leads to convergence, and the two operators that are initialized to translation stays very much unchanged (except the columns of A that fall inside the image region buffer are zeroed out. We believe this is due to the manifold penalty, where pixel transformations inside the buffer still contribute to the manifold penalty function). We can then conclude from this experiment that the learning objective is non-convex and is hard to get to a stable solution when initialized at random. Initialize the model with translation places the algorithm closer to the minimum and helps learning converges.

3.4.5 An Illustration of Learned Models

While most of the learned transformations are hard to make sense of, the model provides better motion estimation as the number of operators increases. This is also evident in the visual quality of the reconstruction. While it is hard to judge the image quality of a 9×9 image patch, we picked two frames from the *Foreman* sequence and show the reconstruction of the entire frame using different learned models. Figure 3.18 shows two frames in the sequence that contain fairly large interframe changes.

Figure 3.18: Frame 13 and 14 of the Foreman sequence.

To code the image, each frame is divided into 17×17 blocks with an overlap of 4 pixels. The blocks in frame 13 are used to estimate blocks in frame 14 with different transformation models. The result is shown in figure 3.19, 3.21, 3.20. As we can see, both the visual quality

and the PSNR increases as the model gets more complex.

Figure 3.19: Reconstruction of frame 14 with motion model consists of only translation operators.

Figure 3.20: Reconstruction of frame 14 with motion model consists of 15 learned operators.

Figure 3.21: Reconstruction of frame 14 with motion model consists of 1, 2, 3 and 4 learned operators in additional to two pre-coded translation operators.

3.5 Conclusion and Future Work

In this chapter we have shown that the proposed learning and inference algorithm perform as we expected. In particular, the inference algorithm is evaluated using pairs of affine transformed images. The algorithm is able to correctly recover 80% of the transformation coefficients. In addition, we have shown that the algorithm is able to correctly learn the operators when trained on affine transformed images. We believe that both of these results are novel and have not been attempted before with similar experimental settings.

Another novelty in this thesis is the learning of transformations from natural videos. We have shown that when learned on pairs of patches of natural movie, the learned operators resembles functionality such as intensity scaling, contrast enhancement, translation and local deformation. The finding differs from previous work, where translation is the only transformation that was discovered. Moreover, we have shown that motion estimation using the learned model performs better both qualitatively and quantitatively when the number of operators available to the learning algorithm increases.

Perhaps the most interesting question raised in this chapter is what are the functionality of the learned transforms. While we have characterized a handful of learned operators, the function of a large portion of the operators remain unknown. It would be very useful to develop methods that is able to systematically analyze the learned operators. More importantly, one should be able to theoretically design these operators so that they can be applied in more general settings with different image patch size.

Chapter 4

Video Coding with Learned Transformations

4.1 Overview

A flexible and accurate motion estimation algorithm has countless applications. One such application is video coding. In the current video coding standard H.264/MPEG-4 AVC, frame-to-frame motion is estimated and compensated with a rigid translational model. However, the dynamics in natural video resulted from complicated object interactions and a simple block-based translation model is insufficient to capture the changes between video frames. Richer motion models are needed to describe such dynamics. The continuous transformation model developed in previous chapters is an example of such model. In particular, we showed in chapter 3 that the learned models are qualitatively better than the rigid translational model.

In this chapter, I propose to improve the current video coding method by replacing the rigid translational motion model with motion models learned from natural videos. A hurdle one may encounter is that while the learned motion models provide a better description of the underlying motion, they have more parameters. This makes the learned model more expensive to encode. The question is how to trade off between the coding cost of motion model and the motion estimation error and how this influences the final bit rate of the compressed video. In this chapter, I analysis this trade off in detail in the context of a hybrid video coder.

4.2 Background

4.2.1 H.264/MPEG-4 AVC

The current video coding standards are developed and maintained by two independent groups, namely the **ITU-T Video Coding Experts Group (VCEG)** and **ISO/IEC Moving Picture Experts Group (MPEG)** . The VCEG standard is original developed to distribute video content over the same channels used to carry telephone communication. In most part of the world, this limits the transfer bit-rate to a multiple of 64kbit/s (e.g. 64kb/s, 128kb/s, etc), depending on the specific transmission protocol. In other words, the VCEG standard facilitates low bit rate video transfer applications and a low end-to-end delay is often desired. In contrast, the primary goal of the MPEG standard was to standardize a compression system capable of delivering higher quality video and audio than that offered by the VCEG standard, where the end-to-end delay is not a key consideration. In recent years, there is a growing demand for very low bit-rate applications such as internet video streaming, mobile phone streaming and video conferencing. To facilitate such needs, MPEG worked together with VCEG and developed a partnership effort known as the Joint Video Team (JVT). The VCEG's H.264 and MPEG's MPEG-4 AVC (Advanced Video Coding) standard are jointly maintained so that they contain identical content.

In the H.264/MPEG-4 AVC standard, videos are coded using a hybrid coding scheme. In particular, the temporal redundancy is exploited with block-based translational motion estimation. The motion prediction error, also known as the motion compensated residual, is then coded using block-based Discrete Cosine Transform (DCT). The total coding cost is the sum of the coding cost of the motion vectors (coefficients of the motion models) and the coding cost of DCT coefficients of the motion compensated residual.

4.2.2 Local and Global Motion Model

The dynamics of video result from a combination of camera motion, object motion and object interaction. One sensible approach to encode video is therefore to factor the video into different objects and model the motion of each object. For example, Han and Woods [Han and Woods, 1998] have utilized more complicated affine motion models for object-based video coding. In their work, the motion parameters are estimated such that they lead to an efficient representation of the motion field inside each segmented region. The difficulty of such methods is that the motion model parameters and the object segmentation needs to be simultaneously estimated and often leads to complicated optimization scheme.

Taking one step back, Josawa et al [Jozawa *et al.*, 1997] proposed to only model the changes of

the background using an affine motion model. The affine motion model is able to model background changes due to camera motion and focal length changes and this allows the authors to exploit long-term statistical dependencies in the video. Because the background motion often changes slowly with respect to the foreground objects, the global affine transform coefficients needs only to be updated sparsely over the length of the entire video sequence. After compensating for global motion, local deformation is estimated using block-based translational model similar to those used in H.264/MPEG-4 AVC. Because the global motion is already compensated, the motion field due to local deformation is likely to be sparse. Improving on this work, Weigand et al [Wiegand *et al.*, 2005] uses several sets of coefficients to model multiple independently moving objects in combination with camera motion and focal length changes. This function is now supported by the MPEG-4 standardization group [Subgroup, 1997].

While affine transformation is used to model global motion, there are a few attempts that used affine model in a block-based setting. Tsai and Huang [Tsai and Huang, 1981] is among one of the earliest attempts that used a parametric motion model to relate motion of planar objects in the scene to observed motion field. However, the parameter estimation accuracy is strongly influenced by the presence of noise in the corresponding feature points.

4.2.3 Coding Motion Compensated Residual with Wavelets and Overcomplete Dictionaries

The motion compensated residuals in H.264/MPEG-4 are coded frame by frame independently with block-based DCT. This is possibly inspired by the success of JPEG image coding standard, which uses almost exactly the same algorithm. Following the success of JPEG2000, where block-based DCT is replaced by wavelets for image coding, one obvious line of work is to use 2D wavelets to encode motion compensated residuals [Zhang and Zafar, 1992]. Unfortunately, the improvement is only marginal comparing to block-based DCT.

Instead of using orthonormal wavelets, Neff and Zakhor [Neff and Zakhor, 2002] showed that it is possible to improve video coding gain at very low bit rate using an overcomplete dictionary consists of oriented Gabor-like filters. The authors used Matching Pursuit to encode the motion compensated residual sparsely where the first few coefficients is able to capture most of the residual content. This type of coder is desirable for ultra-low bandwidth applications such as video streaming on mobile devices.

4.2.4 Video Coding with Spatio-temporal Wavelets

So far we have only considered the hybrid coding scheme, where the spatial and temporal redundancies of time-vary image sequences are coded independently. Several attempts have been made to encode these properties simultaneously, as these properties of video are intimately connected. One of the earlier works done along this line is the use of separable spatio-temporal wavelets. While one can simply apply the spatiotemporal filter blindly over the entire video sequence, Ohm [Ohm, 1994] proposed a way to improve coding gain by first estimate the global motion within a space-time block using a translational motion model. The frames are then wrapped and aligned along the motion trajectories such that most of the signal energy lies in the lower frequency wavelets, resulting a very sparse decomposition of the video. Rahmoune et al [Rahmoune *et al.*, 2006] further proposed to use an overcomplete set of spatio-temporal filter elements instead of separable space-time wavelets.

Using space-time wavelets, Secker and Taubman [Secker and Taubman, 2004] derived a space-time scalable coder using spatio-temporal wavelets, where the resolution in space and time can be delivered incrementally based on the channel bandwidth.

4.3 Block-based Exhaustive Search Motion Estimation

The motion model used in H.264/MPEG-4 is based on rigid translational model. In particular, this type of motion models divide a given video frame into blocks (the basic version divides the frame into 8×8 blocks while the latest standard uses a pyramid search scheme where the largest block size is 32×32). These blocks are known as macroblocks in video compression jargon. For each macroblock $b_N(x, y)$ at position (x,y) in frame N, the algorithm search exhaustively in the previous frame N-1 within an area centered at (x,y) to find the best matching block in the least square sense. In other words, given two image patches with fixed vertical and horizontal translation operators, the algorithm finds the optimal transformation coefficients by performing an exhaustive search over all discretized pixel positions. This process is depicted in figure 4.1.

Sub-pixel interpolation up to a quarter of a pixel is often used to improve the accuracy of the match. Despite being a crude motion model, it is popular within the video compression community partly because it is very efficient to compute. Specifically, the exhaustive search can be implemented using a 2D convolution and many of the image/video processing chips has designated hardware to perform such computation.

The block-based exhaustive search motion model establishes the baseline for this research and all the models used in this chapter will be compared against it.

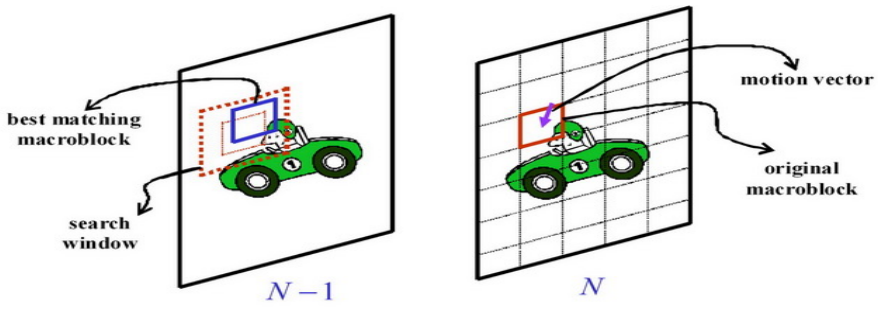

Figure 4.1: An illustration of block based exhaustive search motion estimation.

4.4 Proposed Video Coder Outline

We propose to evaluate different motion models with a realistic coding scheme. Its block diagram is outlined in figure 4.2.

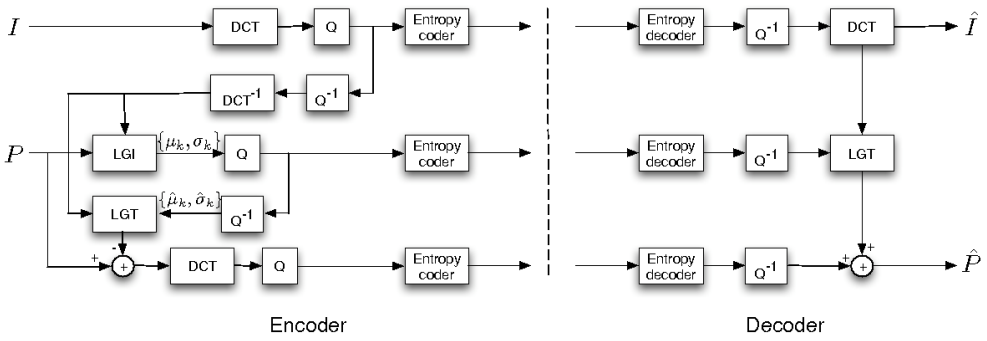

Figure 4.2: Predictive hybrid video coder based on Lie Group transforms.

In this video coder, frames in the image sequence are divided into two types:

- Intra-frame (I-frame). The I-frames are coded without any motion prediction. On the encoder side, the I-frames are divided into non-overlapping 9×9 blocks and each block is independently transform coded with DCT. Each DCT coefficient is uniformly quantized into 8 bits and entropy coded before transmission.
- Inter-frame (P-frame). The P-frames are encoded with a combination of block-based motion prediction from the previous (reference) frame and the quantized prediction residue. In this experiments, the encoder divides each P-frame into non-overlapping

9×9 blocks and motion estimation is performed independently on each block. The motion model coefficients are quantized and the motion compensated residual is evaluated as a difference between the original frame and the estimation obtained using the quantized model coefficients. The motion compensated residual is divided into 9×9 non-overlapping blocks and each block is coded and quantized independently with DCT. All the quantized coefficients are entropy coded before transmission.

In this setup, the motion model coefficients are the set of μ_k and σ_k found using equation 2.8, where $k \in [1, K]$ for K be the number of operators in the continuous transformation model. This is denoted as LGI (Lie Group Inference) in the block diagram. The inverse process, where the image patches are reconstructed given the quantized coefficients, is denoted by LGT (Lie Group Transform).

On the decoder side, the received bit-stream is entropy decoded and de-quantized. The coefficients are inverse transformed to recover the original signal. I-frames are reconstructed from the decoded DCT coefficients while the decoded motion predictions and the motion compensated residues are summed to produce the final P-frames.

The main goal of this chapter is to perform motion estimation with learned transformation models. We investigate how quantization effect the quality of motion estimation and how different motion model influences the coding cost of motion compensated residue. The result is then compare against the traditional exhaustive search motion estimation. For simplicity, we use a I-P-I-P... coding scheme in this chapter.

4.5 Coding Motion Model Coefficients

For the exhaustive search motion estimation model operating on 8×8 blocks, every motion coefficient is 3 bits at full pixel resolution. Even at quarter pixel resolution, every motion coefficient can be coded with 5 bits. In contrast, every coefficient in the continuous transformation model is 32 bits. This posts a great disadvantage for our model when used for video coding. In this section, I attempt to reduce the number of bits used to represent the coefficients via quantization, a process to approximate a larger set of values by a relatively smaller set of values. Unavoidably, quantization introduces error to the coefficients and the quality of motion estimation reduces. The goal of this section is to understand how quantization error relates to the quality of motion estimation and also to the compressibility of the motion coefficient. We also derive analytically the relationship between quantization error and reconstruction quality at the end of this section.

4.5.1 Quantization Schemes

Given the number of bits available to the quantizer, the goal of quantization is to minimize the distortion introduced to the signal. In this section, we investigate the following quantization techniques and compare their performances.

- Uniform quantization. As its name suggests, the continuous number line is divided into equal size bins and all the number falls within a given bin is represented by the median of the bin (some uses the smallest value of the bin). This is illustrated on the left of figure 4.3.

  Uniform quantizers are often found in compression algorithms due to their simplicity. For example, H.264/MPEG 4 AVC uses uniform quantization to quantize the motion compensated residual. More importantly, the uniform quantization is near optimal in the least square sense when the underlying continuous distribution does not form multiple clusters, i.e. when there is no 'gaps' in the underlying distribution. Having gaps in the distribution results empty bins, which is wasteful and means that the quantization scheme can be improved by moving the empty bin somewhere else.

- Deadzone quantization. It is the same as the uniform quantizer, except that the zero bin(the bin of which its quantized value is 0), is twice as large as the other bin. This is illustrated on the right of figure 4.3.

  Similar to uniform quantization, deadzone quantization works well when the underlying continuous distribution does not form multiple clusters. In addition, the deadzone quantizer is designed for distribution that has a strong peak at 0. For example, it is used in JPEG2000 for quantize the wavelet coefficients. The deadzone quantizer also assumes that quantization error is proportional to the reconstruction error. Since the contributions of the near zero coefficients to reconstruction error are small, it is able to trade off signal quality with compressibility.

- Lloyd-Max quantization. Given the number of desired bins, the Lloyd-Max quantizer finds the optimal bin-widths and bin-centroids that minimizes signal distortion in the least square sense [Lloyd, 1982] . It is essentially the K-mean clustering algorithm applied on scalar variables. It is mostly used when the underlying distribution is complicated or unknown. The drawback is that training is needed to find the quantization parameters.

The uniform and deadzone quantizer can be formulated mathematically as

$$\bar{x} = \left\lfloor \frac{x}{\Delta} + \tau \right\rfloor \tag{4.1}$$

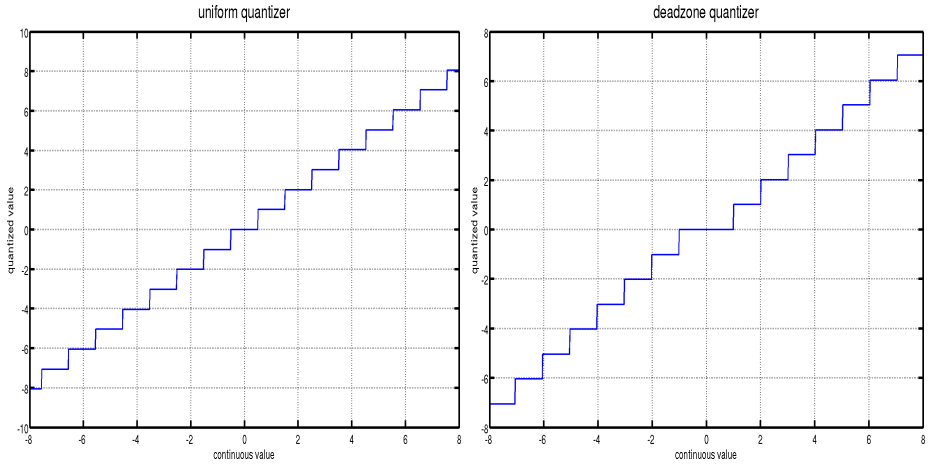

Figure 4.3: Uniform quantizer (left) and deadzone quantizer (right).

where x is the continuous variable, $\bar{x}$ is the quantized variable and Δ is the bin size. τ governs the size of the zero bin, for which it is an uniform quantizer when $\tau = 0.5$ and the deadzone is 2Δ when $\tau = 0$.

4.5.2 Distortion in Quantized Coefficients

In this section, we determine the relationship between reconstruction quality and compressibility empirically. We study this relationship for each of the quantizer type listed above so that we can find the most suitable one for this application. Figure 4.4 shows the average PSNR of the reconstruction of P-frames for the first 40 frames of *Foreman* sequence in a I-P-I-P coding scheme. The motion model is a continuous translation model (see appendix B for detail) applied on 9×9 blocks with a 8 pixel search radius around each side.

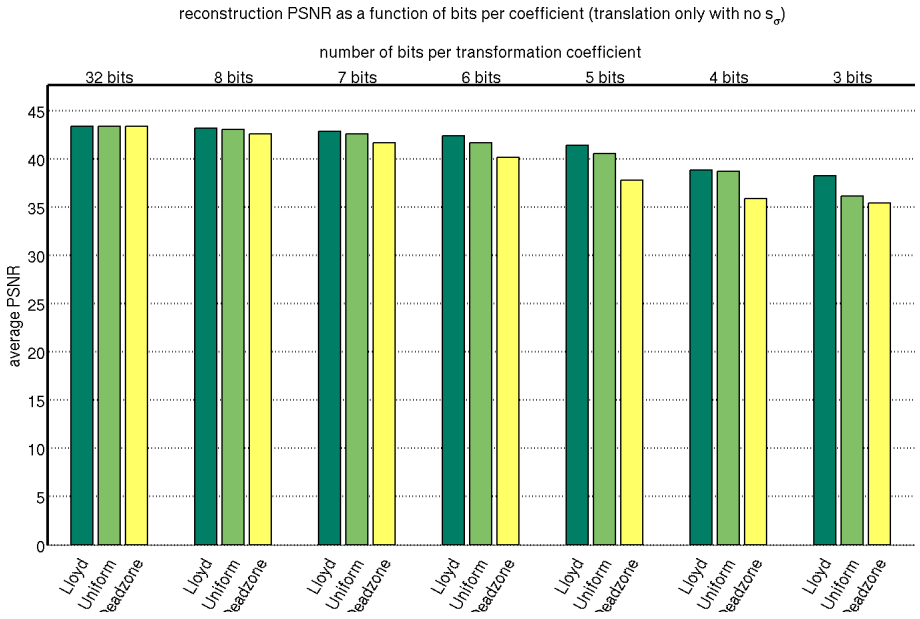

Figure 4.4: The 9×9 block average PSNR of motion estimation of P-frames with handcoded translation operators without s_σ of the first 40 frames of the *Foreman* sequence. The three different quantizers are uniform, deadzone and Lloyd-Max.

The first observation is that by quantizing the coefficients into 8 bits, there is no significant reduction in PSNR of the model reconstruction (0.2dB for Lloyd-Max and 0.3dB for uniform). For an uniform quantizer, the quantization error for a 8 bit quantizer is approximately 0.04 of a pixel. Given that the highest resolution traditional motion estimation uses is quarter of a pixel, this result should not be surprising. The uniform quantizer shows a 1.5dB PSNR reduction when the coefficients are quantized into 6 bits. This translates to a 0.14 pixel error. On the other hand, the Lloyd-Max quantizer only showed a 0.7 dB degradation when quantized use 6 bits. This shows that the Lloyd-Max quantizer is much better in capturing the information at lower bit rate. The justification for this can be seen by looking at the underlying continuous motion coefficient distribution. In figure 4.5 we plotted the coefficient histogram of a vertical translation operator where the coefficients are inferred using a continuous translation operator without s_σ variables.

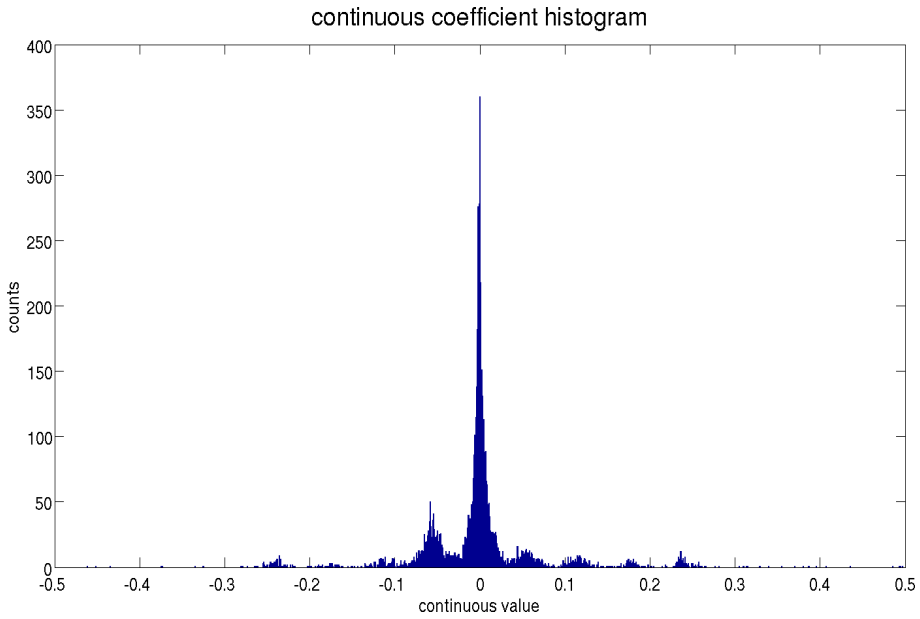

Figure 4.5: The histogram of the vertical translation operator coefficients inferred with a translation only model with no s_σ. The histogram is collected over the P-frames of the first 40 frames of *foreman*.

As shown, the distribution of the coefficient is multi-modal. This agrees with the energy landscape of the objective function, shown in figure 2.1. The exact reason for why the algorithm prefer some shift values over others is unknown and would be a topic for future work. In the case of an uniform quantizer, the bins are lay down evenly over the entire range and some bins are wasted when put in places that has a small histogram count. The Llody-Max algorithm however places the bin at the peaks of the histogram and therefore is more efficient. This is shown in figure 4.6.

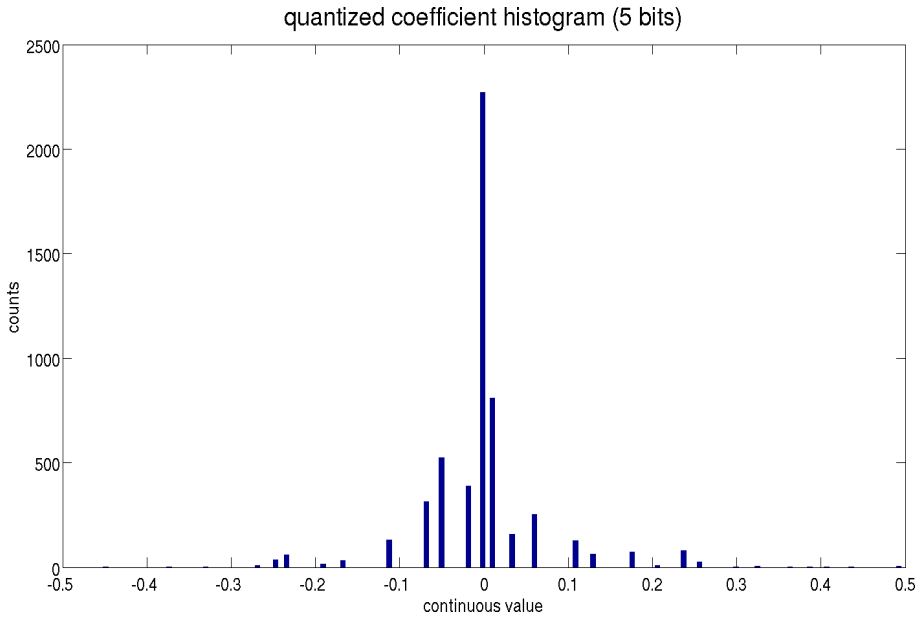

Figure 4.6: The histogram of the vertical translation operator coefficients quantized into 5 bits using Lloyd-Max quantization.

Moreover, the bin width of the Lloyd-Max quantizer is approximately proportional to the bin centroid, i.e. the bin width is small for small coefficient values and bigger for bigger coefficient values. This is evident in figure 4.7 and is a direct reflection of the sparse nature of the underlying coefficients.

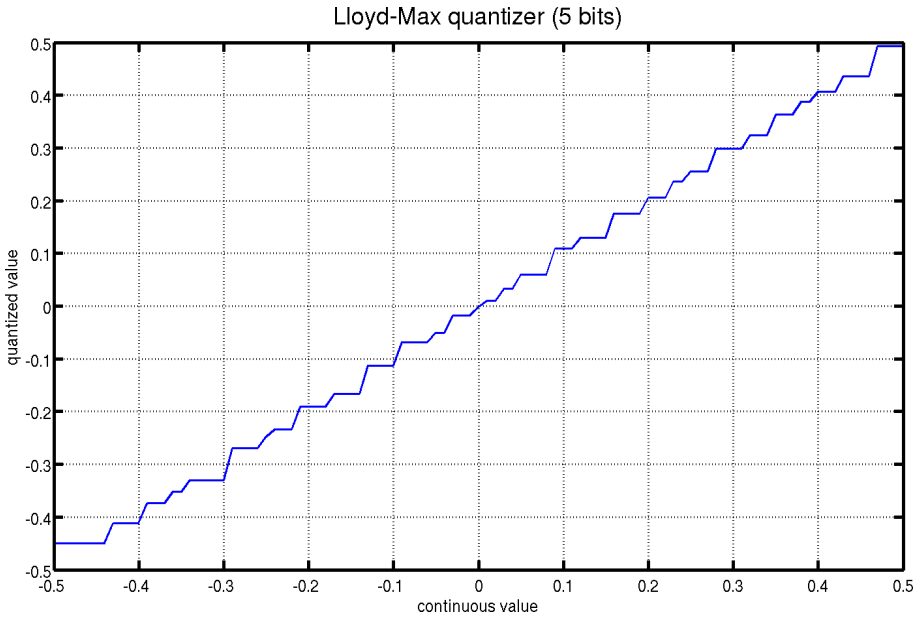

Figure 4.7: The quantization learned by the Llody-Max algorithm to quantize vertical translation operator coefficients into 5 bits. The coefficients are inferred with a translation only model with no s_σ on the P-frames of the first 40 frames of *foreman*.

This also explains the poor performance of deadzone quantizer in lower bit regime. The major assumption in deadzone quantizer is that the reconstruction error is proportional to the coefficient values, which is the case for all linear models. However, the model used here is nonlinear and reconstruction error is not proportional to coefficient values. As an example, for translation, the error resulted by translating an image by 1 pixel can be as large as the error resulted by translating the image by 2 pixels. This can be also seen from the objective function in figure 2.1. Therefore, given the underlying coefficient is heavily peaked around zero, quantizing all the smaller values to exactly zero causes a large error in reconstruction.

Similar quantization error behavior is also observed in the learned transformations, shown in figure 4.8.

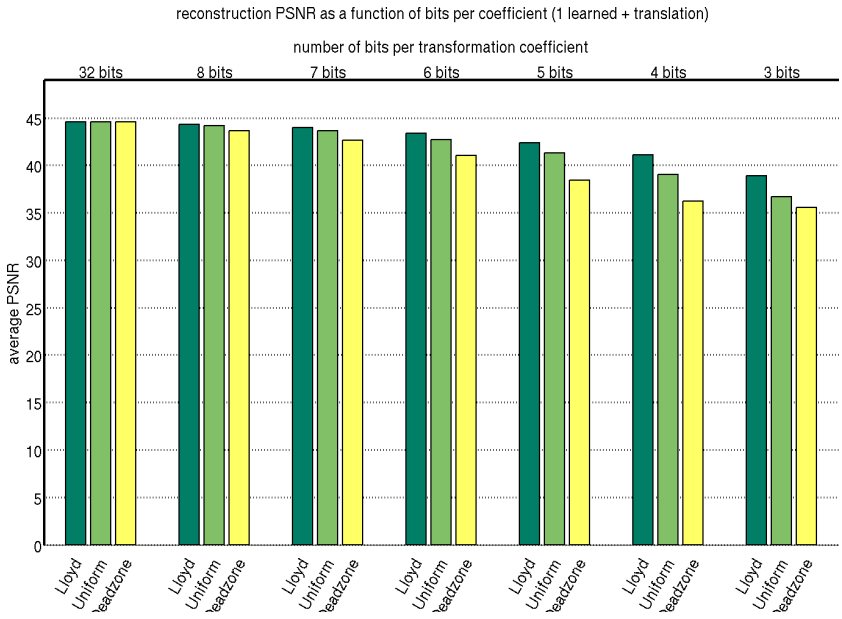

Figure 4.8: The 9×9 block average PSNR of motion estimation of the P-frames with handcoded translation operators + 1 learned transformation of the first 40 frames of the *Foreman* sequence. The s_σ values are kept at 32 bits and only the s_μ are quantized in this setting. The three different quantizers are uniform, deadzone and Lloyd-Max.

Now that we have compared reconstruction quality of different quantization scheme, we need to compare the compressibility of the quantized coefficients. It is obvious that the deadzone quantizer results a more compact representation compare to the Lloyd-Max quantizer. The compressibility of a quantized symbol stream can be calculated using its entropy. The entropy of a symbol stream is defined as follows:

$$H(X) = \sum_k p(x_k) \log_2 \left(p(x_k)\right) \tag{4.2}$$

where $H(X)$ is the number of bits required to code each symbol in the symbol stream. In figure 4.9, we show the total number of bits needed to code the quantized coefficients versus the PSNR of the motion model reconstruction with the quantized coefficients. This curve is often known as the rate distortion curve in the signal processing literature.

The optimal quantizer would lie on the top left corner of the the rate distortion curve, which correspond to high PSNR with low bit rate. In general, the bit rate increases as the PSNR increases and the rate distortion curve allows one to inspect the PSNR of a given rate (and vice versa) for different models. As shown in the figure, the Lloyd-Max quantizer behaves

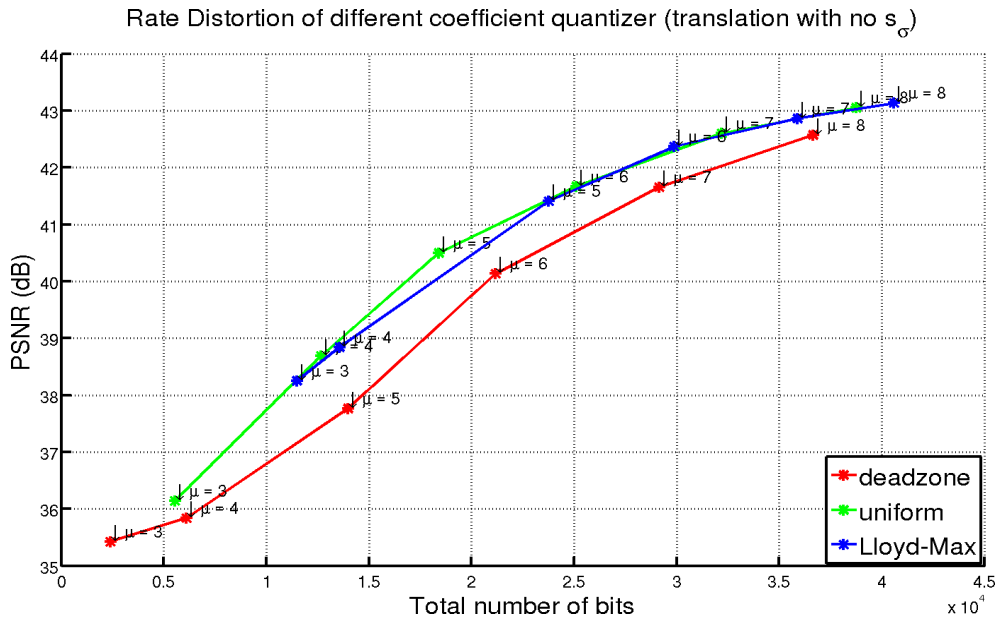

Figure 4.9: The rate distortion curves of the translation only model without s_σ evaluated on the P-frames of the first 40 frames of the *foreman* sequence. Each point on the curve corresponds to a different coefficient quantization, ranging from 3 bits to 8 bits. The average PSNR is computed over all the non-overlapping 9×9 blocks. The three different quantizers are uniform, deadzone and Lloyd-Max.

very similar to the uniform quantizer at high bit rate (5 bits or more) while performs slightly worse at lower bit rate. On the other hand, the Lloyd-Max quantizer is able to reach a higher PSNR at maximum bit rate. Judging from this plot, it is safe to say that uniform quantization should be used for low bit rate while Lloyd-Max quantization should be used for high bit rate.

For the rest of this chapter, we will consider both the uniform quantization and Lloyd-Max quantization such that we can compare the model performance in both low and high bit rate regime.

4.5.3 Relationship between s_μ and s_σ

In the continuous transformation model, the s_μ and s_σ variables represent different properties of the transform and the distortion behavior is expected to be different under quantization. To investigate this, we plotted the rate distortion curve for the handcoded translation model

in figure 4.10 with different quantization of s_μ and s_σ. The coefficients are quantized using the Llody-Max quantizer.

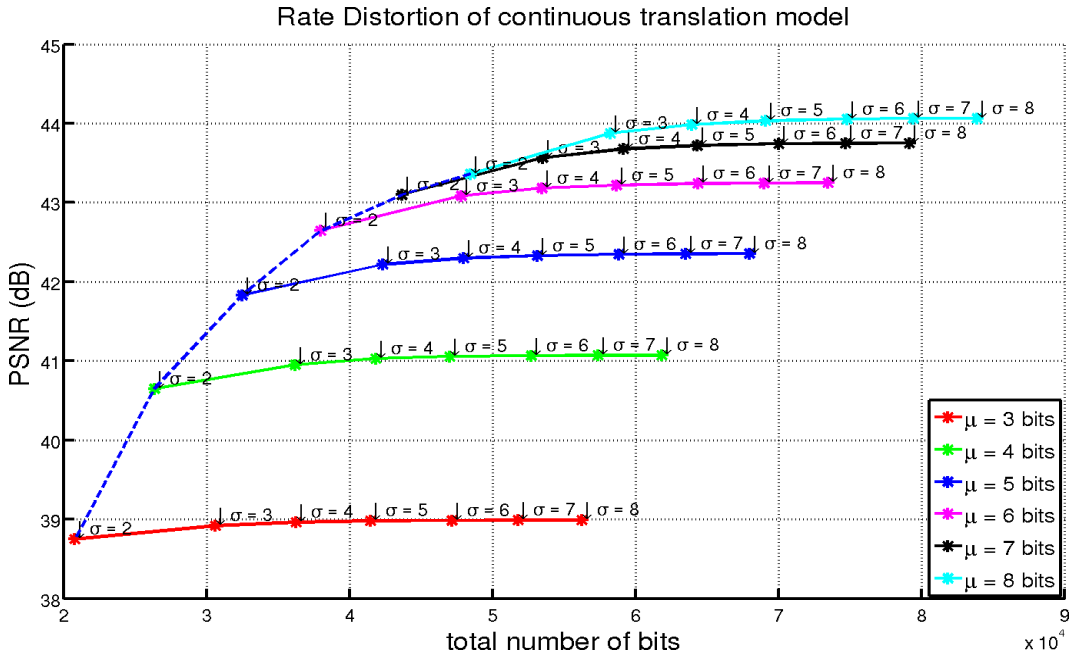

Figure 4.10: The rate distortion curves of the translation only model evaluated on the P-frames of the first 40 frames of the *foreman* sequence. The average PSNR is computed over all the 9×9 blocks. Each individual curve is a fixed quantization of s_μ with varying quantization of s_σ. The dotted blue line represents the outer hull of the rate distortion curves.

First, the s_μ variables are much more sensitive to the quantization error than the s_σ variables. In particular, the for a given s_μ quantization, the quantization error for s_σ plateaus after approximately 4 bits. There is very little gain from quantizing s_σ beyond 4 bits. In contrary, every bit increase for the s_μ coefficients cause a large jump in PSNR. More importantly, the rate distortion characteristics of the continuous translation model, represented by the outer hull of the set of rate distortion curves, plotted as the dotted blue line, can be largely captured by changing the precision of s_μ. This means that if we have a limited resources, it is better to vary the precision in s_μ rather than s_σ. Similar observation is found for the learned transformation model too, shown in figure 4.11.

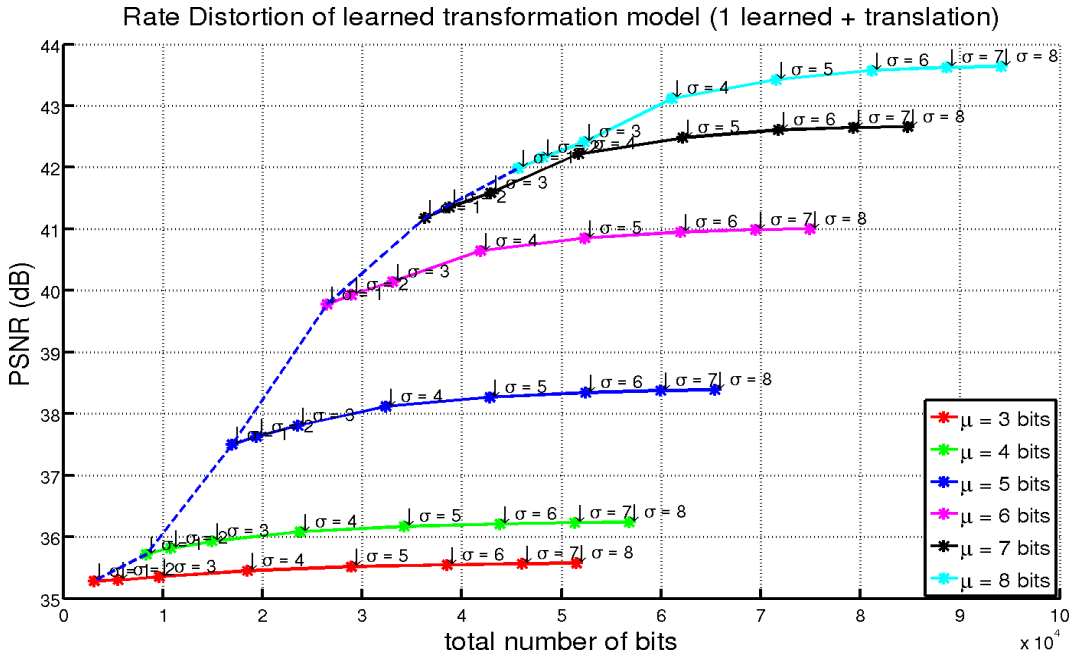

Figure 4.11: The rate distortion curves of the translation model plus 1 learned operator evaluated on the P-frames of the first 40 frames of the *foreman* sequence. The average PSNR is computed over all the 9×9 blocks. Each individual curve is a fixed quantization of s_μ with varying quantization of s_σ. The dotted blue line represents the outer hull of the rate distortion curves.

4.5.4 Rate Distortion of Different Motion Models

Using the rate distortion curves, we compare the coding efficiency of different motion models, shown in figure 4.12. In the figure, the series of rate distortion functions of a given motion model resulted from changing the quantization level of s_μ and s_σ is presented by their outer hull. In this experiment, the s_μ are quantized between the range of 3 bits to 8 bits and the s_σ are quantized from 2 bits to 6 bits. All combinations are explored. In this figure, we also showed the rate distortion curve of the exhaustive search motion estimation with sub-pixel accuracy. The five points on the curve represent full pixel, half pixel, quarter pixel, one-eighth and one-twelfth pixel accuracy.

First, we can see that the exhaustive search motion estimation is more efficient in the low bit regime. While disappointing, the result is not too surprising because its motion vectors do not suffer from quantization error, as the motion vectors themselves are already discrete. Therefore, this may pose an advantage over the continuous transformation models, where the inference algorithm is ignorant about the quantization distortion. One plausible future work would be to incorporate quantization inside the inference algorithm.

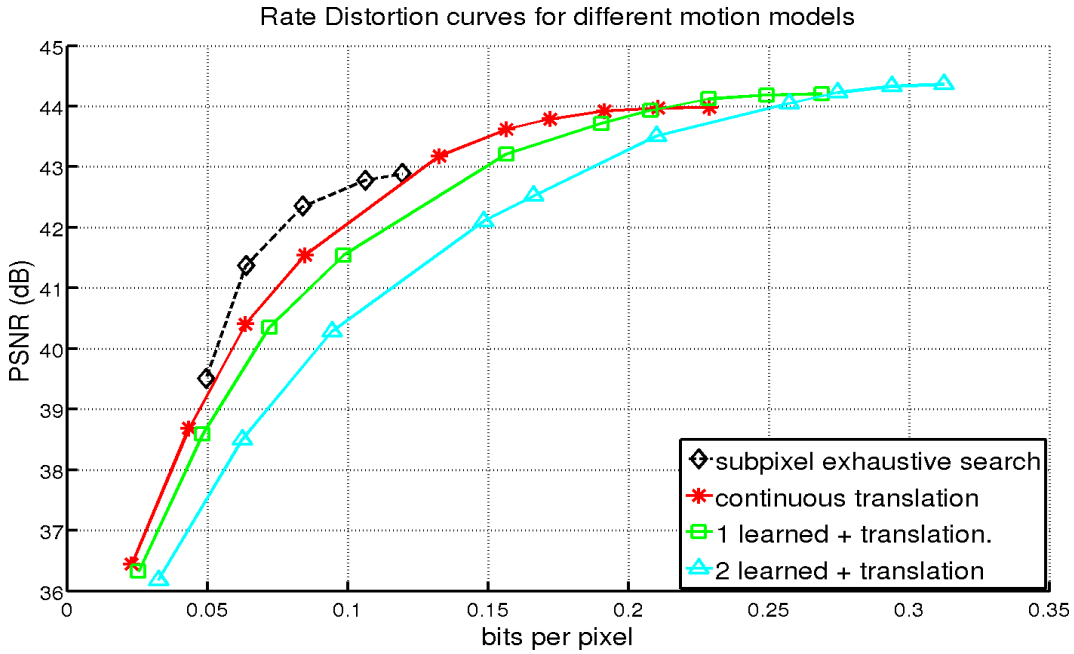

Figure 4.12: The rate distortion of the P-frame of the first 40 frames of *foreman* with different motion models. The average PSNR is computed over all the 9×9 blocks. The motion model coefficients are quantized using uniform quantization.

While the exhaustive search algorithm triumphs at low bit rate, it plateaus at approximately 43dB. On the other hand, the continuous transformation models are able to reach 44dB or more. This gives them an advantage at higher bit rate. Finally, at very low bit rate, the continuous transformation models are also more efficient than the exhaustive search motion model.

4.5.5 Analytical Distortion Analysis

In this section, we attempt to derive the relationship between signal reconstruction error and quantization error analytically. First, the total distortion between the original image patch $\mathbf{Y}$ and the quantized reconstructed image patch $\hat{\mathbf{Y}}$ can be lower bounded by the triangle inequality:

$$D_{tot} = \|\mathbf{Y} - \hat{\mathbf{Y}}\|^2 \leq \|\mathbf{Y} - \tilde{\mathbf{Y}}\|^2 + \|\tilde{\mathbf{Y}} - \hat{\mathbf{Y}}\|^2 = D_a + D_q, \tag{4.3}$$

where D_a is the distortion due to the approximation error in the inference process, D_q is the distortion due to quantization of the transform coefficient. For algebraic clarity, we will develop the quantization-distortion relationship for $\mathbf{s}_\mu$ only. However, a relationship can be

derived for $\mathbf{s}_\sigma^2$ as well by following a nearly identical derivation.

Let $\hat{\mathbf{T}}$ be the transform using the quantized coefficient $\hat{\mathbf{s}_\mu}$, $\hat{\mathbf{T}} = \mathbf{T}(\hat{\mu}, \sigma)$. For simplicity, we will denote s_μ to be μ and s_σ to be σ. The distortion D_q due to quantization error can be written as

$$D_q = \|\mathbf{T}\mathbf{x} - \hat{\mathbf{T}}\mathbf{x}\|^2 = \|(\mathbf{T} - \hat{\mathbf{T}})\mathbf{x}\|^2 = \mathbf{x}^\mathsf{T}\Delta\mathbf{T}^\mathsf{T}\Delta\mathbf{T}\mathbf{x}. \tag{4.4}$$

Using the multiple transformation equation in 2.9, we get:

$$\Delta\mathbf{T} = \prod_{k=1}^{K} \mathbf{U}_k e^{\mu_k \mathbf{\Lambda}_k} e^{\frac{1}{2}\mathbf{\Lambda}_k^2\sigma_k^2}\mathbf{U}_k^{-1} - \prod_{k=1}^{K} \mathbf{U}_k e^{\hat{\mu}_k \mathbf{\Lambda}_k} e^{\frac{1}{2}\mathbf{\Lambda}_k^2\sigma_k^2}\mathbf{U}_k^{-1} \tag{4.5}$$

Writing $e^{\hat{\mu}_k\mathbf{\Lambda}_k}$ as a Taylor expansion around $e^{\mu_k\mathbf{\Lambda}_k}$, we have

$$e^{\hat{\mu}_k\mathbf{\Lambda}_k} = e^{\mu_k\mathbf{\Lambda}_k} + \Delta\mu_k\mathbf{\Lambda}_k e^{\mu_k\mathbf{\Lambda}_k} + \dots\,, \tag{4.6}$$

where $\hat{\mu}_k = \mu_k + \Delta\mu_k$, i.e., $\Delta\mu_k$ is the quantization error for the transformation coefficient μ_k. Since $\Delta\mu$ is typically small, especially in medium and high rate regimes, we can approximate $\Delta\mathbf{T}$ using only the zeroth and first order terms. We substitute back into Eq. 4.5, expand the product, and drop the higher order terms in μ,

$$\begin{aligned}
\Delta\mathbf{T} &= \prod_{k=1}^{K} \mathbf{U}_k e^{\mu_k \mathbf{\Lambda}_k} e^{\frac{1}{2}\mathbf{\Lambda}_k^2\sigma_k^2}\mathbf{U}_k^{-1} - \prod_{k=1}^{K} \mathbf{U}_k \left[e^{\mu_k \mathbf{\Lambda}_k} + \Delta\mu_k\mathbf{\Lambda}_k e^{\mu_k\mathbf{\Lambda}_k} + \dots\right] e^{\frac{1}{2}\mathbf{\Lambda}_k^2\sigma_k^2}\mathbf{U}_k^{-1} \\
&= \prod_{k=1}^{K} \mathbf{T}_k\left(\mu_k, \sigma_k\right) - \prod_{k=1}^{K} \mathbf{T}_k\left(\mu_k, \sigma_k\right) \\
&\quad - \sum_{l=1}^{K} \left(\prod_{k=1}^{l-1} \mathbf{T}_k\left(\mu_k, \sigma_k\right)\right) \mathbf{U}_l \Delta\mu_l \mathbf{\Lambda}_l e^{\mu_l\mathbf{\Lambda}_l} e^{\frac{1}{2}\mathbf{\Lambda}_l^2\sigma_l^2}\mathbf{U}_l^{-1} \left(\prod_{k=l+1}^{K} \mathbf{T}_k\left(\mu_k, \sigma_k\right)\right) \\
&\quad - \sum_{l=1}^{K}\sum_{m=1}^{K} \dots \\
\Delta\mathbf{T} &\approx \sum_{l=1}^{K} \Delta\mu_l \mathbf{M}^l
\end{aligned}$$

where

$$\mathbf{M}^l = -\left(\prod_{k=1}^{l-1} \mathbf{T}_k\left(\mu_k, \sigma_k\right)\right) \mathbf{U}_l \mathbf{\Lambda}_l e^{\mu_l\mathbf{\Lambda}_l} e^{\frac{1}{2}\mathbf{\Lambda}_l^2\sigma_l^2}\mathbf{U}_l^{-1} \left(\prod_{k=l+1}^{n} \mathbf{T}_k\left(\mu_k, \sigma_k\right)\right) \tag{4.7}$$

This gives us a quadratic form for the quantization distortion D_q,

$$D_q \approx \Delta\mu^T \mathbf{Q} \Delta\mu, \tag{4.8}$$

where the coupling matrix $\mathbf{Q}$ is quadratic in $\mathbf{x}$,

$$Q_{kl} = \mathbf{x}^T \left(\mathbf{M}^{k^T} \mathbf{M}^l \right) \mathbf{x}. \tag{4.9}$$

4.6 Coding Motion Compensated Residual

In the previous section, we have shown that a more complex motion model is often more expensive to code. The justification for using such a model in the video compression context is that a richer motion model is able to better model the changes between video frames. This can be seen in figure 4.13, where we plotted the PSNR of the video reconstruction of the P-frames of the *foreman* sequence with different motion models using unquantized coefficients.

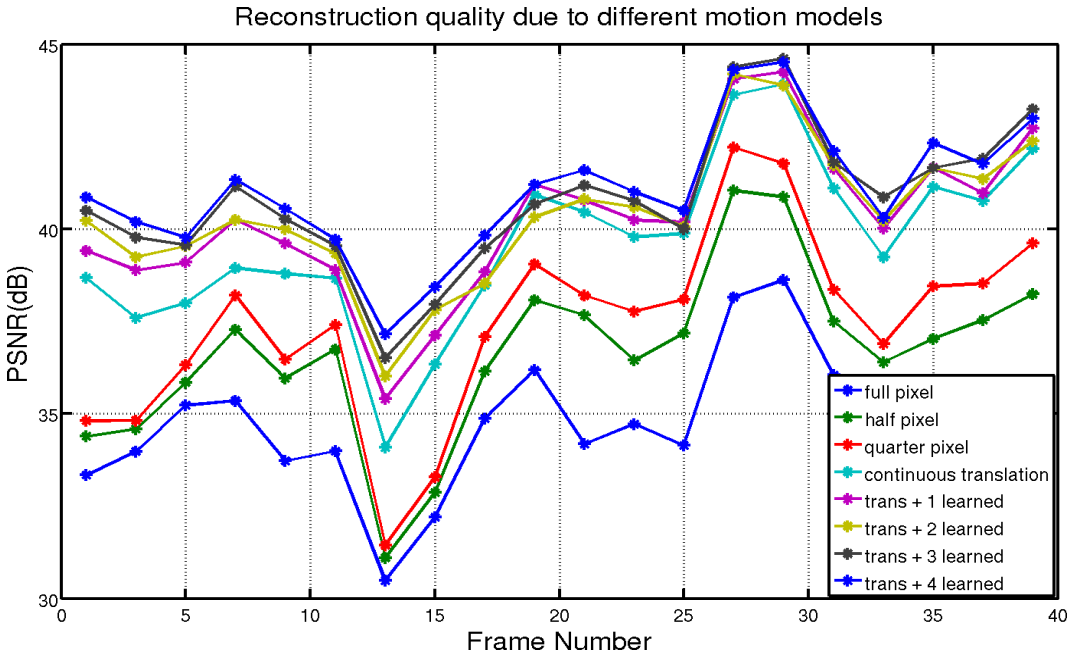

Figure 4.13: The PSNR of the reconstruction of each individual frame of the P-frames of the first 40 frames of *foreman* using different motion models with unquantized coefficients.

As we can see, each additional operator (2 additional coefficients per macro-block) results a 0.5 - 1 dB PSNR increase. We can also see that the PSNR gap between the exhaustive search motion model could be as large as 3dB compare to even the continuous translation model. This quantitative difference can be also seen qualitatively. In figure 4.14 and 4.15, we show the motion compensated residue in the pixel and frequency domain for different motion models.

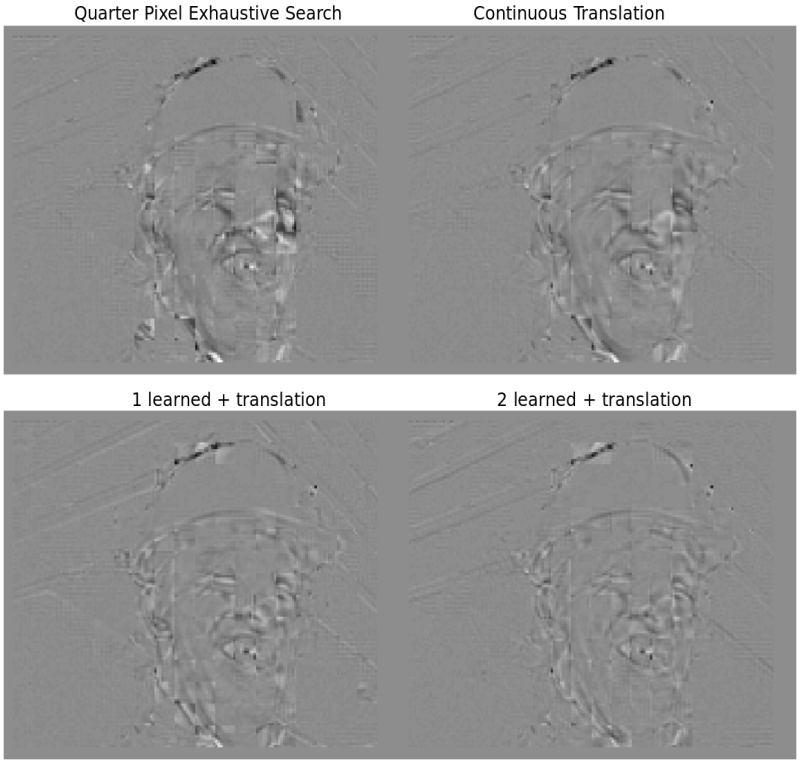

Figure 4.14: The motion compensated residue of P-frame 14 of $foreman$ computed with different motion models. All the sub-figures are normalized to the same scale.

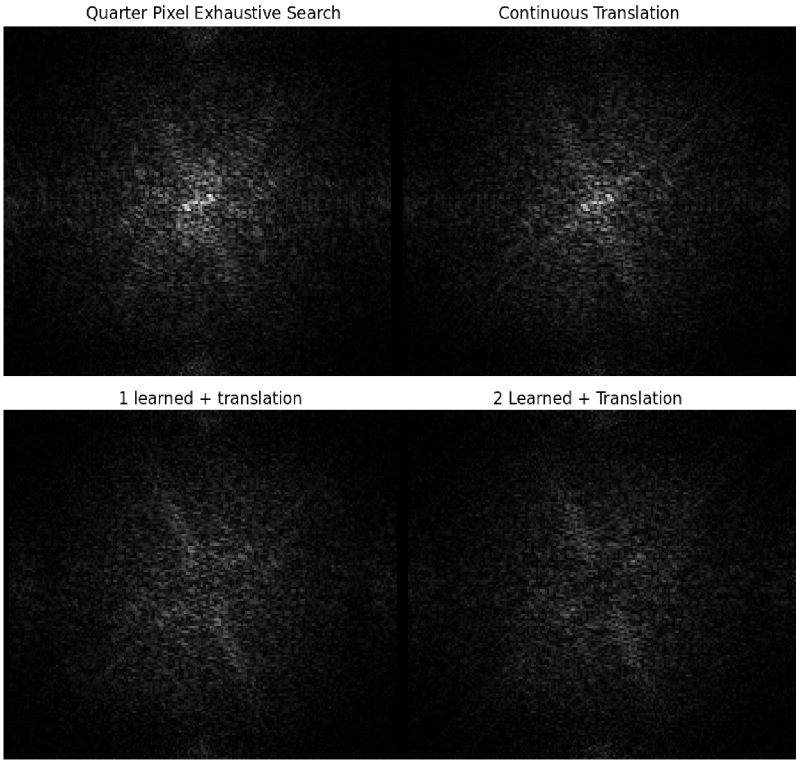

Figure 4.15: The power spectrum of the motion compensated residue of P-frame 14 of *foreman* computed with different motion models. All the sub-figures are normalized to the same scale.

Evidently, the total power in the motion compensated residue reduces as the number of operators in the transformation model increases. In addition, there are less visible structures in the pixel domain as the motion models become more complex. The power distribution over difference frequencies is also flatter. This shows that the residue approaches the noise distribution. More importantly, the learned operators mainly account for motion with lower spatial frequencies as power in the lower frequency band decrease significantly with additional operators. This is understandable because the power in natural scene is inversely proportional to the frequency. When the learning algorithm is penalized by a sum of square error, it is most essential for the model to correctly reconstruct the lower frequency content.

As we have discussed earlier, the total coding cost of a given video in a hybrid coding scheme is the sum of coding cost of the model and the coding cost of the residue. The motivation for using a more complicated motion model is that a richer motion model results better approximation and hence lower residual energy. Therefore while the coding cost of motion model is higher, the coding cost for the residue may be lower and thus the total cost may be lower. In particular, while there are two additional coefficients for one additional operator in

the motion model, there are a total of 81 coefficients for the residue in a 9×9 macro-block. The coding cost for the residue therefore seems to dominate the coding cost of the video. In fact, previous work [Secker and Taubman, 2004] has pointed out that the motion model often contributes to a small faction of the total coding cost in a high rate regime. In this section, we are going to investigate this.

4.6.1 Residual Coding Strategy

In H.264/MPEG-4 AVC, the motion compensated residue is coded per frame with a spatial transform. In the earlier standards, the residue frame is divided into 8×8 blocks and continuous valued DCT is used to code each block. The continuous DCT coefficients are quantized, in which the older VCEG standards (H.261 and H.262) quantized the coefficients using uniform quantization while MPEG quantized the coefficients using a quantization table. Such quantization table quantizes the lower frequencies much finer than the higher frequencies and is derived from the human contrast sensitivity function.

The more recent H.264/MPEG-4 AVC standard uses a 4×4 DCT-like invertible integer transform. The advantage of using such a transform is that the coefficients are integer valued and hence does not subject to quantization error when transmitted as is. Furthermore, the computation required for this transformation is much smaller than the continuous value DCT transform and is preferred for real-time compression applications.

In this chapter, we will investigate the following residual coding strategies

- 9×9 Discrete Fourier Transform. DCT is used to code natural image patches (JPEG) because it closely resembles the structure of the principle components of natural images while being efficient to compute. Specifically, principle components represent the optimal directions in the lease square sense for dimensionality reduction.

- 9×9 Eigen Vector Transform. The Eigen vectors of the motion compensated residue is computed and the residues are projected onto the Eigen vectors for which the coefficients are used for coding. The Eigen vectors represents the principle components of the residual and when the distribution of the residue deviates away from natural images, the Eigen vectors would be more efficient in dimensionality reduction than DCT.

- 4×4 integer valued DCT used in H.264/MPEG4-AVC.

4.6.2 Rate Distortion of Residual

Firstly, we would like to compare the performance of the motion compensated residual coding methods listed above. To do so, a continuous translation model is used to estimate the motion in the P-frames of the first 40 frames of *foreman*. The s_μ and s_σ parameters in the model are quantized into 6 bits and 4 bits respectively. The resulting motion compensated residue is then coded using one of the three methods listed above, and each of the transform coefficients are quantized into different bits using uniform quantization. Figure 4.16 shows the rate distortion curve due to the residue coding only.

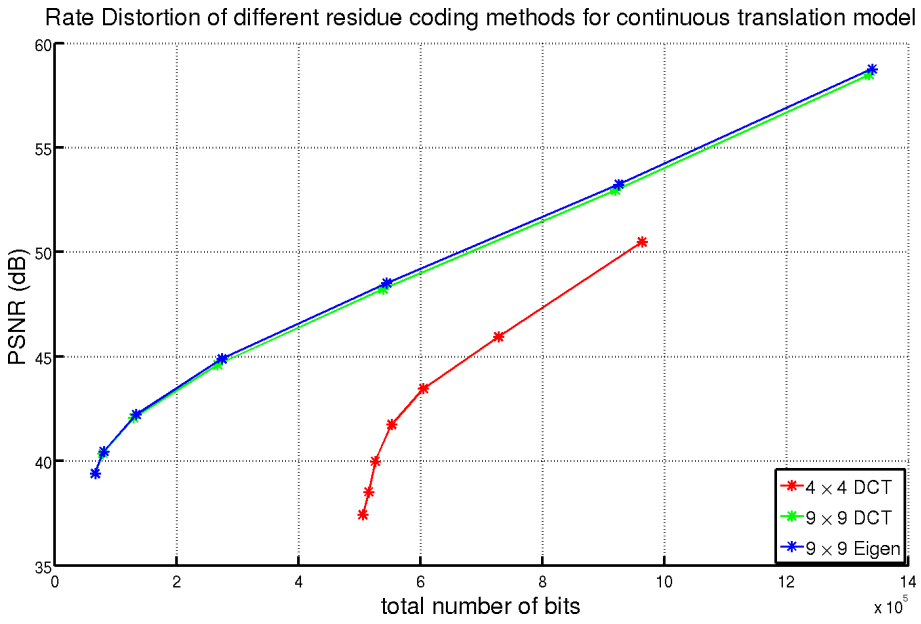

Figure 4.16: The rate distortion curve of the residue using a translation only model. The s_μ and s_σ are quantized into 6 bits and 4 bits respectively. The residue is coded with 9×9 DCT, 9×9 eigen vectors and 4×4 integer value DCT.

As we can see, the performance for the 9×9 DCT and 9×9 Eigen decomposition is almost the same, with the Eigen decomposition performs slightly better than the DCT. This result is unexpected because the structure contained in the Eigen vectors, shown in figure 4.17, is rather different than DCT. However, similar rate distortion result is also observed in three operator and 4 operator models. This shows that the rate distortion curve is insensitive to the particular basis used.

On the other hand, the 4×4 integer value DCT performs poorly. This is possibility due to the fact that the motion estimation is applied on 9×9 image patches and therefore there

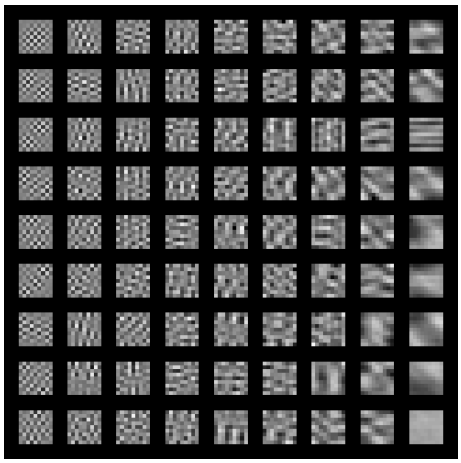

Figure 4.17: 81 Eigen vectors obtained by applying principle component analysis on 9×9 blocks for motion compensated residual extracted from the P-frames of the first 40 frames of *foreman*. The continuous translation model is used.

exists motion artifact across the 4×4 blocks. This effect is not persistent in H.264/MPEG-4 because it uses 8×8 blocks. As part of the future work, all the operators should be learned on 8×8 blocks.

4.7 Rate Distortion of the Hybrid Coder

In this section, we will consider the coding efficiency of the entire hybrid coder. In order to gain insight of how the different parameters (s_μ, s_σ and residue coefficients) in the model interact, we plot the rate distortion of the video using the translation only model fixing the quantization of either s_μ and s_σ. In particular, from figure 4.10, we know that the PSNR saturates for the quantization of s_σ more than 4 bits. For simplicity, we will fix s_σ to 4 bits. Figure 4.18 shows the rate distortion of the continuous translation model with varying s_μ and residue quantizations.

One can immediately notice that the s_μ parameters play a major role in the performance of the video coder. More importantly, only the smaller quantization of the residue contributes to the outer hull of the rate distortion function.

Finally we consider the coding efficiency of different learned motion models. This is shown in figure 4.19. In the graph, the lines represents the outer hull of the rate distortion of each

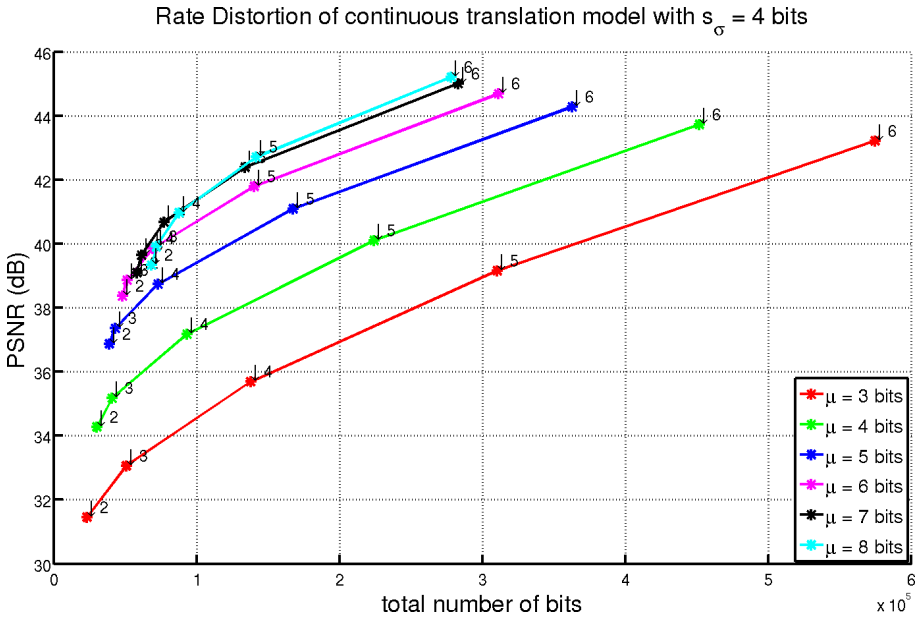

Figure 4.18: The rate distortion curves of the P-frames of the first 40 frames of *foreman* encoded with translation only model. For each given quantization of s_μ, the motion compensated residue is quantized into 2 to 6 bits. The computation of PSNR is based on the entire image. The s_σ coefficients are quantized into 4 bits.

model.

As shown, the continuous translation model outperforms the traditional exhaustive search motion estimation method in the low rate and high rate regime. The continuous translation model performs better than the exhaustive search algorithm at higher rate because it is able to achieve a higher PSNR. It outperforms the exhaustive search algorithm at low bit rate because s_σ can be quantized coarsely while deliver a boost in PSNR. Unfortunately, we are unable to show that the more complex models with learned operators are more effective in terms of coding than the simpler handcoded motion models.

4.8 Conclusion and Future Work

In this chapter we illustrated video coding using learned motion models in a hybrid coding scheme. Different methods for quantizing and coding the motion model coefficients and the motion compensated residues are considered and compared. We showed that the continuous translation model outperforms the exhaustive search motion compensation model used in current video coding standard H.264/MPEG-4 AVC in the low and high bit rate regime.

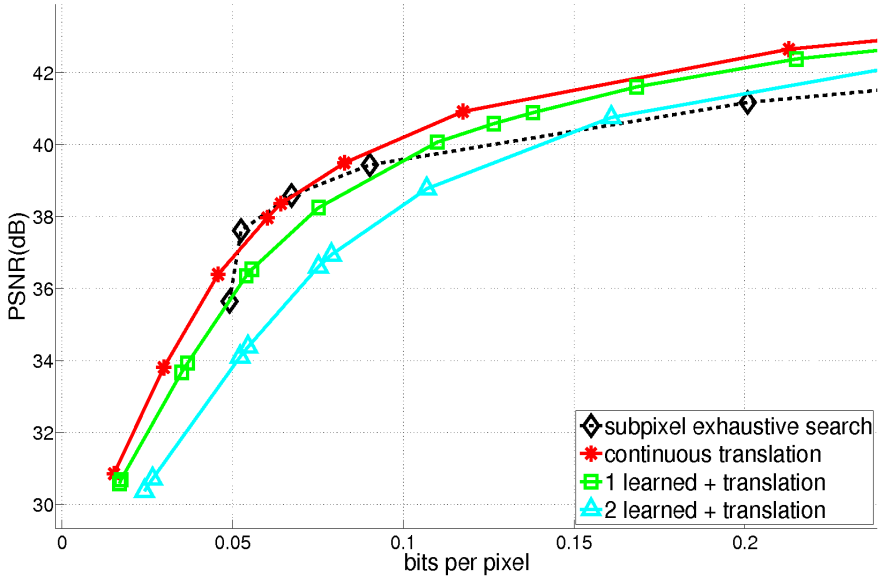

Figure 4.19: The rate distortion curves of the P-frames of the first 40 frames of *foreman* encoded with different motion models.

There are lots of room for improving the proposed video coding framework. We outline some of them in the following:

- The proposed inference and learning algorithm models motion in each video frame independently. The algorithm ignores the fact that the same motion often extends over multiple frames in natural video. To model such long range dependencies, we can augment our model to learn and infer on a group of frames. One particular implementation can be done with the following

$$f(t) = T(ts_\mu, ts_\sigma)f(0) \tag{4.10}$$

  where $t \in [0, T]$, for which T is the number of frames in the group. In particular, the single set of coefficient s_μ, s_σ is common between all the frames in the group and is inferred together. The time index t is used to model the transition between frames. In doing so, we implicitly assume that the changes between frames within the group is constant.

Such model can be extremely beneficial for video coding since only one set of parameters is needed to model the entire group of frames. This also allow us to perform motion interpolation and prediction between video frames.

- As we have pointed out earlier, block-based DCT may not be the optimal coding strategy for the learned motion models, because the motion compensated residue become more sparse in the pixel domain as the motion model becomes more complex. Therefore, one possible alternative that is able to take advantage of the residue structure is to use full-frame coding with a per-coded/learned overcomplete dictionary, such as the coding framework used by Neff and Zakhor [Neff and Zakhor, 2002].

- In the current coding framework, the motion model and residue model are considered separately. The result would be better if the motion model and the residue model is learned and inferred together. Specifically, one could consider the following generative model

$$Y = T(s_\mu, s_\sigma)X + \Phi\alpha + |\alpha|_0 \tag{4.11}$$

  where $\Phi \in \Re^{M\times N}$ and $\alpha \in \Re^{N\times 1}$. In the model, the coefficients of the motion compensated residue is encouraged to be sparse through a L0 penalty.

- The Llody-Max quantizer optimizes the quantization parameter by minimizing the distortion between quantized coefficient and the true coefficient. However, minimizing this distortion does not guarantee that the image reconstruction error is also minimized, as the two are nonlinearly related. One way to improve this is to optimize quantization parameter by directly minimizing distortion in the image domain using equation 4.8. For further improvement, one can factor the quantization process into the inference and learning scheme.

Chapter 5

Contributions

In this chapter I summarize the major contributions of this thesis and discuss the implications of these contributions.

Continuous Transformation Model

Regarding the continuous transformation model, we

- derived an unsupervised learning algorithm that computes the model update of an exponential transformation model in polynomial time.
- derived an adaptive inference rule that is able to avoid local minima.
- demonstrated that the inference algorithm can simultaneously recover multiple transformation parameters with high accuracy.
- demonstrated that the algorithm is able to learn novel transformation operators when trained on natural videos.
- demonstrated that the learned models provide a better description of the underlying dynamics compare to currently used motion models.

Having an efficient learning rule and an inference rule that avoids local minima allows us to learn the continuous transformation model without using approximations. This differentiates us from previous approaches where they linearize the exponential model to make learning / inference tractable. The benefit of using the exponential model is that it can handle data with all transformation ranges, in contrast to the linearized models, where the approximations only holds for infinitesimal transformations. This feature allows us to successfully apply our model in practice, where the range of transformations is unpredictable.

I am able to show that the algorithm learned interesting transformations such as translation, brightness adjustment and contrast enhancement when learned on natural videos. There are still some fraction of the learned transformations that are uninterpretable, but nevertheless their contribution can be quantified. One line of work in the future would be to develop methods to analyze these operators.

In this thesis, I demonstrated inference and learning on natural video data. However, there are a lot of other areas where transformational relationship in data can be exploited. For example, in hand-written characters, the same character can be written in many different ways. To recognize them, it is important to tease apart the identity of the character and the writing style. Current recognition system more or less ignores the style and simply assign different appearance of the same character to the same label. A better alternative is to model the transformation between data so that we can tease apart the style and the content. Other possible applications include video interpolation and view interpolation.

Video Coding with Learned Transformations

Regarding the video coding framework, we

- described a hybrid coding scheme that replaces the traditional exhaustive search motion estimation model with the continuous transformation model using learned operators.
- investigated the distortion of the continuous motion models due to quantization error.
- demonstrated that the continuous transformation model outperforms the traditional motion model at low and high bit regime.

This work proposes an alternative to the exhaustive search motion model for video coding, which has been in use for the last thirty years. As discussed in chapter 4, a relatively small amount of work has been done on finding better motion models for video coding. Part of this is due to the fact that there is no satisfactory method for describing and parameterizing block based motion. We hope our work here provides a plausible path for the video coding community to pursuit.

There are a lot of improvements that can be made to the current implementation. To start, we currently only model frame-to-frame transformations. We expect that the coder would be more efficient if we consider a group of frames and exploit long range dependencies among them. In addition, we believe that the quantization step can be factored into the inference and learning algorithm. In this way, the quantization error can be minimized.

Appendices

Appendix A

Objective Function Derivatives

Let

$$\varepsilon = \sum_n (Y_n - U e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2 \sigma_s^2} U^{-1} X_n)^2$$

A.1 Derivative for inference with one operator

The learning gradient with respect to μ_s is

$$\begin{aligned}\frac{\partial \varepsilon}{\partial \mu_s} &= -\sum_n 2err(n) U \frac{\partial e^{\mu_s \Lambda}}{\partial \mu_s} e^{\frac{1}{2}\Lambda^2 \sigma_s^2} U^{-1} X_n \\ &= -\sum_n 2err(n) U \Lambda e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2 \sigma_s^2} U^{-1} X_n\end{aligned}$$

where $err(n)$ is the reconstruction error of the nth sample

$$err(n) = Y_n - U e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2 \sigma_s^2} U^{-1} X_n$$

Similarly, the learning gradient with respect to σ_s is

$$\frac{\partial \varepsilon}{\partial \sigma_s} = -\sum_n 2err(n) U e^{\mu_s \Lambda} \sigma_s \Lambda^2 e^{\frac{1}{2}\Lambda^2 \sigma_s^2} U^{-1} X_n$$

A.2 Derivative for learning with one operator

The learning gradient with respect to Λ is

$$\frac{\partial \varepsilon}{\partial \Lambda} = -\sum_n 2err(n)U\frac{\partial e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2\sigma_s^2}}{\partial \Lambda}U^{-1}X_n$$

It is easy to see then this derivative with respect to each element in Λ is

$$\frac{\partial \varepsilon}{\partial \lambda_k} = -\sum_n 2err(n)U(\mu_s e^{\mu_s \lambda_k} + \sigma_s^2 \lambda_k e^{\frac{1}{2}\Lambda^2\sigma_s^2})\ U^{-1}X_n$$

Therefore, in matrix form, the derivative is

$$\frac{\partial \varepsilon}{\partial \Lambda} = -\sum_n 2err(n)U(\mu_s e^{\mu_s \Lambda} + \sigma_s^2 \Lambda e^{\frac{1}{2}\Lambda^2\sigma_s^2})\ U^{-1}X_n$$

The learning gradient with respect to U is

$$\begin{aligned}\frac{\partial \varepsilon}{\partial U} = &-\sum_n 2err(n)\frac{\partial U}{\partial U}e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2\sigma_s^2}U^{-1}X_n \\ &-\sum_n 2err(n)Ue^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2\sigma_s^2}\frac{\partial U^{-1}}{\partial U}X_n\end{aligned}$$

Recall that

$$\frac{dU^{-1}}{dU} = -U^{-1}\frac{dU}{dU}U^{-1}$$

The learning gradient is hence

$$\begin{aligned}\frac{\partial \varepsilon}{\partial U} = &-\sum_n 2err(n)e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2\sigma_s^2}U^{-1}X_n \\ &+\sum_n 2err(n)Ue^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2\sigma_s^2}U^{-1}U^{-1}X_n\end{aligned}$$

A.3 Derivative for Complex Variables

To accommodate for the complex variables U and Λ, we rewrite our objective function as

$$\varepsilon = \sum_n err(n)^T \overline{err(n)}$$

where $\overline{err(n)}$ denotes the complex conjugate. The derivative of this error function with respect to any complex variable can be then broken into the real and and imaginary part

$$\begin{aligned}
\frac{\partial \varepsilon}{\partial \Lambda} &= \sum_n err(n)^T \overline{\frac{\partial err(n)}{\partial \Lambda}} + \frac{\partial err(n)^T}{\partial \Lambda} \overline{err(n)} \\
&= \sum_n err(n)^T \overline{\left(\frac{\partial err(n)}{\partial \Lambda}\right)} + \left(\frac{\partial err(n)}{\partial \Lambda}\right)^T \overline{err(n)} \\
\Re\left\{\frac{\partial \varepsilon}{\partial \Lambda}\right\} &= 2\Re\left\{\sum_n err(n)^T \overline{\left(\frac{\partial err(n)}{\partial \Lambda}\right)}\right\} \\
\Im\left\{\frac{\partial \varepsilon}{\partial \Lambda}\right\} &= -2\Im\left\{\sum_n err(n)^T \overline{\left(\frac{\partial err(n)}{\partial \Lambda}\right)}\right\}
\end{aligned}$$

A.4 Derivatives in Matrix Notation

For completeness, we can write the derivatives in matrix notation as follows.

$$\begin{aligned}
\frac{\partial \varepsilon}{\partial \mu_s} &= -2E^T U \Lambda K U^{-1} X \\
\frac{\partial \varepsilon}{\partial \sigma_s} &= -2\sigma_s E^T U \Lambda^2 K U^{-1} X \\
\frac{\partial \varepsilon}{\partial \Lambda} &= -2E^T (\mu_s e^{\mu_s \Lambda} + \sigma_s^2 \Lambda e^{\frac{1}{2}\Lambda^2 \sigma_s^2})\ U^{-1} X \\
\frac{\partial \varepsilon}{\partial U} &= -2E^T K U^{-1} X + 2E^T U K U^{-1} U^{-1} X
\end{aligned}$$

where

$$K = e^{\mu_s \Lambda} e^{\frac{1}{2}\Lambda^2 \sigma_s^2}$$

E and X are matrices with columns of err(n) and X_n respectively.

Appendix B

Affine Transformation Operators

Affine transformation refers to a linear transformation of the vector space followed by a translation:

$$x' = Ax + b \tag{B.1}$$

In image transformation, x and x' are typically image coordinates. A total of seven different transformation can be described under this framework:

- horizontal and vertical translation
- rotation
- horizontal and vertical scaling
- horizontal and vertical sheer

B.1 Coordinate-wise Affine Transformation Matrices

Let x and y be the image coordinates before the transform and x' and y' be the coordinates after the transform.

Translation in X direction

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ 0 \end{bmatrix}$$

Translation in Y direction

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ t_y \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling in X direction

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling in Y direction

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Skew in X direction

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Skew in Y direction

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

After the coordinate-wise transform, the image is created using bilinear interpolation.

B.2 Analytical operator for translation under proposed model

The translation operator under our proposed model can be formulated analytically. To remind the readers, our proposed model is

$$I_1 = Ue^{s_\mu \Lambda} U^{-1} I_0$$

On the other hand, one can perform translation in the Fourier domain as follows

$$FT[I(x - x_0, y - y_0)] = e^{-i2\pi\omega_x x_0} e^{-i2\pi\omega_y y_0} FT[I(x, y)] \tag{B.2}$$

where FT denotes Fourier Transform, x_0 and y_0 are real numbers denoting the amount of translation in x and y direction respectively. Therefore, to perform translation, one would take the image, compute its Fourier transform, scale the phase of the complex amplitude according to equation B.2, and compute its inverse Fourier transform. Therefore, to equate our model to equation B.2, we can easily see that

- U^{-1} is the Fourier transform basis of one direction
- U is the inverse Fourier transform basis of the same direction
- Λ is a diagonal matrix which its nth entry is $-i2\pi n$

Figure B.1 shows the horizontal and vertical translation operator created by the method above.

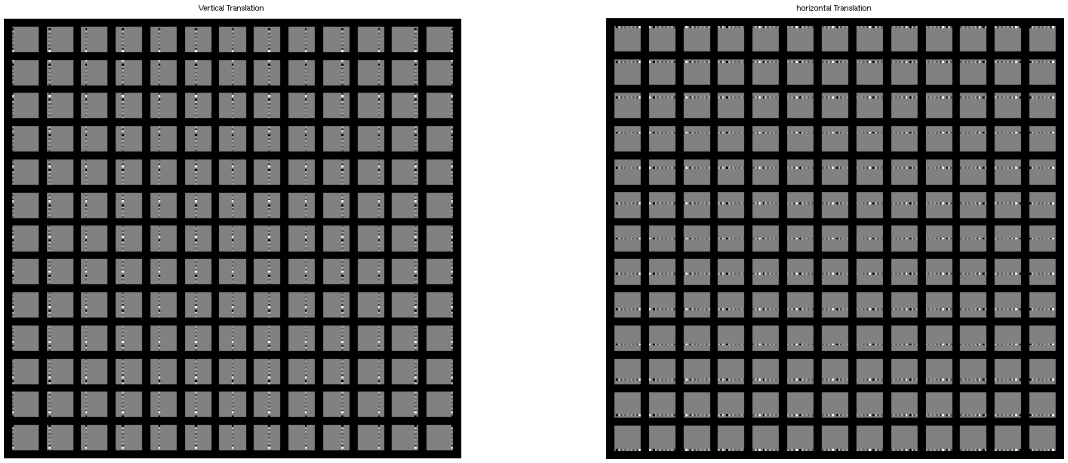

Figure B.1: Vertical (left) and Horizontal (right) translation operator

Bibliography

[Adelson and Bergen, 1985] E.H. Adelson and J.R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of Optical Society of America A*, 2:284–299, 1985.

[Barlow, 1989] H. B. Barlow. Unsupervised learning. *Neural Computation*, 1(1):295–311, 1989.

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

[Black and Jepson, 1996] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Proc. of the Fourth European Conference on Computer Vision*, (ECCV):320–342, 1996.

[Cadieu and Olshausen, 2009] C. Cadieu and B.A. Olshausen. Learning transformational invariants from natural movies. *Neural Information Processing Systems (NIPS)*, 21:209–216, 2009.

[Culpepper and Olshausen, 2009] Benjamin Culpepper and Bruno Olshausen. Learning transport operators for image manifolds. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 423–431. 2009.

[Escoda *et al.*, 2009] Oscar D. Escoda, Gianluca Monaci, Rosa F. Ventura, Pierre Vandergheynst, and Michel Bierlaire. Geometric video approximation using weighted matching pursuit. *IEEE Transactions on Image Processing*, 18(8):1703–1716, 2009.

[Fahle and Poggio, 1981] M. Fahle and T. Poggio. Visual hyperacuity: spatiotemporal interpolation in human vision. *Proceedings of the Royal Society London B*, 213:451–477, 1981.

[Friedman *et al.*, 1974] J. H. Friedman, J. W. Tukey, et al. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–890, 1974.

[Han and Woods, 1998] S. C. Han and J. W. Woods. Adaptive coding of moving objects for very low bit rates. *IEEE Journal on Selected Areas in Communications*, 16(1):56–70, 1998.

[Hateren and Ruderman, 1998] J. H. Van Hateren and D. L. Ruderman. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceeding of the Royal Society London B*, 265(1412):2315–2320, 1998.

[Hildreth, 1984] E. C. Hildreth. Computation underlying the measurement of visual motion. *Artificial Intelligence*, 23:309–355, 1984.

[Hinton and Salakhutdinov, 2006] Geoff E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[Jozawa *et al.*, 1997] H. Jozawa, K. Kamikura, A. Sagata, H. Kotera, and H. Watanabe. Two-stage motion compensation using adaptive global mc and local affine mc. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(2):75–85, 1997.

[Kokiopoulou and Frossard, 2009] E. Kokiopoulou and P. Frossard. Minimum distance between pattern transformation manifolds: Algorithm and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1225–1238, 2009.

[Lloyd, 1982] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):120–137, 1982.

[Lucas and Kanade, 1981] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Understanding Workshop*, 1:121–130, 1981.

[Mahajan *et al.*, 2009] D. Mahajan, F. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. Moving gradients: A path-based method for plausible image interpolation. *ACM Transactions on Graphics (SIGGRAPH 09)*, 28(3), 2009.

[Mairal *et al.*, 2010] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[Mallat and Zhang, 1993] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

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

[Neff and Zakhor, 2002] R. Neff and A. Zakhor. Matching pursuit video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(1):13–39, 2002.

[Nordberg, 1994] K. Nordberg. Signal representation and processing using operator groups. Technical Report 366, Linkoping Studies in Science and Technology, 1994.

[Ohm, 1994] Jens-Rainer Ohm. Three-dimensional subband coding with motion compensation. *IEEE Transactions on Image Processing*, 3(5):559–571, 1994.

[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.*, 2002] B.A. Olshausen, R.P.N. Rao, and M.S. Lewicki. *Probabilistic Models of the Brain: Perception and Neural Function*. MIT Press, 2002.

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

[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.

[Rahmoune *et al.*, 2006] Adel Rahmoune, Pierre Vandergheynst, and Pascal Frossard. Flexible motion-adaptive video coding with redundant expansions. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(2):178–190, 2006.

[Rao and Ruderman, 1999] R.P.N. Rao and D.L. Ruderman. Learning lie groups for invariant visual perception. *Proceedings of Neural Information Processing Systems*, 11:810–816, 1999.

[Roweis and Saul, 2000] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):793–799, 2000.

[Schmidt, 2009] Mark Schmidt. minfunc. Technical report, http://www.cs.ubc.ca/ schmidtm/Software/minFunc.html, 2009.

[Secker and Taubman, 2004] A. J. Secker and David Taubman. 'highly scalable video compression with scalable motion coding. *IEEE transactions on image processing*, 13(8):1029–1041, 2004.

[Serre *et al.*, 2007] T. Serre, A. Oliva, and T. Poggio. A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences (PNAS)*, 104(15):6424–6429, 2007.

[Simoncelli and Heeger, 1998] Eero P. Simoncelli and David Heeger. A model of neuronal responses in visual area mt. *Vision Research*, 38(5):743–761, 1998.

[Smith and Lewicki, 2006] E. C. Smith and M. Lewicki. Efficient auditory coding. *Nature*, 439(7079):800–805, 2006.

[Subgroup, 1997] Video Subgroup. Core experiment on global motion compensation. Technical report, ISO/IEC JTC1/SC29/WG11 MPEG96/M1686, 1997.

[Tsai and Huang, 1981] R. Y. Tsai and T. S. Huang. Estimating three-dimensional motion parameters of a rigid planar patch. *IEEE Transactions on Acoustic and Speech Signal Processing*, 29(6):1147–1152, 1981.

[Van Gool *et al.*, 1995] L. Van Gool, T. Moons, E. Pauwels, and A. Oosterlinck. Vision and lies approach to invariance. *Image and Vision Computing*, 13(4):259–277, 1995.

[Vasconcelos and Lippman, 1997] N. Vasconcelos and A. Lippman. Multiresolution tangent distance for affine invariant classification. *Proceedings of Neural Information Processing Systems*, 10, 1997.

[Watson and Albert J. Ahumada, 1985] Andrew B. Watson and Jr Albert J. Ahumada. Model of human visual-motion sensing. *Journal of the Optical Society of America A*, 2(2):322–342, 1985.

[Wiegand *et al.*, 2003] Thomas Wiegand, Gary Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[Wiegand *et al.*, 2005] Thomas Wiegand, Eckehard Steinbach, and Bernd Girod. Affine multi-picture motion-compensated prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2):197–209, 2005.

[Zhang and Zafar, 1992] Y. Q. Zhang and S. Zafar. Motion-compensated wavelet transform coding for color video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2:285–296, 1992.