UNIVERSITY OF CALIFORNIA

Los Angeles

Learning Inhomogeneous FRAME Models for Object Patterns

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Statistics

by

Jianwen Xie

2014

ABSTRACT OF THE THESIS

Learning Inhomogeneous FRAME Models for Object Patterns

by

Jianwen Xie

Master of Science in Statistics

University of California, Los Angeles, 2014

Professor Ying Nian Wu, Chair

This research investigates an inhomogeneous version of the FRAME (Filters, Random field, And Maximum Entropy) model and apply it to modeling object patterns. The inhomogeneous FRAME is a non-stationary Markov random field model that reproduces the observed marginal distributions or statistics of filter responses at all the different locations, scales and orientations. The experiments show that the inhomogeneous FRAME model is capable of generating a wide variety of object patterns in natural images. It is useful for object detection, alignment, and clustering.

The thesis of Jianwen Xie is approved.

Nicolas Christou

Frederic Paik Schoenberg

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2014

To my parents . . .

for their continual and unconditional support

TABLE OF CONTENTS

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Related work 2

1.3 Organization 2

2 Inhomogeneous FRAME model **3**

2.1 Model and learning algorithm 3

2.1.1 Notation 3

2.1.2 Model 3

2.1.3 Maximum likelihood learning 6

2.1.4 Computing normalizing constants 7

2.2 Simulation by Hamiltonian Monte Carlo 8

2.3 Summary of the learning algorithm 11

2.4 Experiment 1: Synthesis by inhomogeneous FRAME model 11

3 Detection **16**

3.1 Detection algorithm 16

3.2 Experiment 2: Detection by template matching 17

4 Alignment **21**

4.1 Alignment algorithm 21

4.2 Experiment 3: Multiple images alignment 22

5 Clustering **25**

5.1 Mixture of inhomogeneous FRAME models and EM 25

5.2 Model-based k-mean clustering 26

5.3 Experiment 4: Model-based clustering 27

6 Discussion **30**

References **32**

LIST OF FIGURES

2.1 The inhomogeneous FRAME is a generative model that seeks to represent and generate object patterns shown above. (From top to bottom: hummingbird, deer, cat, tiger, and lion). 4

2.2 Synthesized images generated by the inhomogeneous FRAME model. The sizes of the images are 70×70. Each row contains 6 independent samples (synthesized images) drawn from the learned model. 13

2.3 Learning sequence by inhomogeneous FRAME. The size of the images are 70×70. A separate model is learned from each training set shown in Fig 1 (From top to bottom: hummingbird, deer, cat, tiger, and lion). Synthesized images generated in step $t = 1, 7, 10, 20, 50, 100, 200$, and 500. 14

2.4 Images generated by the inhomogeneous FRAME model learned from different categories of objects. The training images are collected from the internet and are cropped so that the training images for each category are roughly aligned. The number of training images for each category is around 10. 15

3.1 Detection (tigers). On the top: A synthesized image generated by the learned model. The rest: Testing images with bounding boxes locating the detected objects. 18

3.2 Detection (wolves). On the top: A synthesized image generated by the learned model. The rest: Testing images with bounding boxes locating the detected objects. 19

3.3 Detection (zebras). On the top: A synthesized image generated by the learned model. The rest: Testing images with bounding boxes locating the detected objects. 20

4.1 Alignment results (deers). Training images with bounding boxes locating the aligned objects. 23

4.2 Cropped objects after alignment (deers) 23

4.3 Alignment results (ducks). Training images with bounding boxes locating the aligned objects. 24

4.4 Cropped objects after alignment (ducks) 24

5.1 EM clustering: On the top: A synthesized image for each cluster is displayed. The rest: 4 clusters of images separated by the model-based EM algorithm. 28

5.2 k-mean clustering. Each row illustrates one clustering example by displaying a synthesized image and a typical training example for each cluster. 29

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Ying Nian Wu, for introducing me to the field of machine learning. He taught me not only the valuable knowledge in machine learning but also how to conduct high-quality research. I am grateful to Professor Song-Chun Zhu, the director of the VCLA (Center for Vision, Cognition, Learning, and Arts), for his valuable advice and supports, and for creating a stimulating research environment that has been very beneficial to me. I am also very grateful to Dr. Wenze Hu for his help in this project. This project would not have been possible without his incredible patience and invaluable help. Thanks to Professor Frederic Paik Schoenberg and Professor Nicolas Christou for taking their time to review my thesis and give me suggestion. My special thanks go out to all my friends just for making my time here at UCLA a pleasure. At last, I thank my parents for all their love, support, and encouragement.

CHAPTER 1

Introduction

1.1 Background and Motivation

Developing generative models for image patterns is one of the most fundamental problems in vision. Although the past decade has witnessed tremendous advance in developing discriminative methods for object recognition, the progress in developing generative models has been lagging behind. The goal of this paper is to develop generative models for object patterns.

The foundation of our work is the FRAME (Filters, Random field, And Maximum Entropy) model that Zhu, Wu, and Mumford (1997) [23] proposed for texture patterns. Being a texture model, FRAME is a spatially stationary Markov random field model, and it is the maximum entropy distribution that reproduces the observed marginal histograms of responses from a band of filters, where for each filter tuned to a specific scale and orientation, the marginal histogram is spatially pooled over all the pixels in the image domain.

In this article, we investigate an inhomogeneous version of the FRAME model for representing object patterns instead of texture patterns. The inhomogeneous FRAME model is a spatially non-stationary random field that reproduces distributions or statistics of filter responses at individual locations, scales and orientations without spatial pooling.

Our experiments show that the inhomogeneous FRAME model are capable of generating realistic object patterns observed in images of natural scenes. Also, the learned

models can be useful for object detection, alignment, and clustering.

1.2 Related work

This class of models are also called energy-based models [18] [1], exponential family models, and Gibbs distributions in various contexts. Examples include the FRAME model [23], field of experts [16], product of experts [8], product of t model [20], restricted Boltzmann machine [17] [9] and its many recent generalizations such as [15] and the references therein. A Markov random field model is defined by an energy function and may involve latent variables or hidden units. If the latent variables are conditionally independent given the observed data or visible units, which is usually the case, the latent variables can be integrated out, resulting in a marginal energy-based model defined by a free energy function.

1.3 Organization

In this paper, we assume that the bank of filters are given, such as Gabor filters and difference of Gaussian (DoG) filters as in the original FRAME model. They can be learned if the training data are abundant.

The rest of the thesis is organized as follows. Section 2 presents the inhomogeneous FRAME model. Section 3, Section 4, and Section 5 study object detection, alignment, and clustering by using inhomogeneous FRAME model. Section 6 concludes with a discussion.

CHAPTER 2

Inhomogeneous FRAME model

2.1 Model and learning algorithm

2.1.1 Notation

We start from modeling roughly aligned images of object patterns from the same category, such as images in Figure 1. Let $\{\mathbf{I}_m, m = 1, ..., M\}$ be a set of training images defined on image domain $\mathcal{D}$. We use the notation $B_{x,s,\alpha}$ to denote a basis function such as a Gabor wavelet centered at pixel x (which is a two-dimensional vector), and tuned to scale s and orientation α ($B_{x,s,\alpha}$ is also an image on $\mathcal{D}$ although it is non-zero only within a local range. We assume that $B_{x,s,\alpha}$ are translated, dilated and rotated versions of each other). We assume that s and α take values within a finite and properly discretized range. The inner product $\langle \mathbf{I}, B_{x,s,\alpha} \rangle$ can be considered the filter response of $\mathbf{I}$ to a filter of scale s and orientation α at pixel x (Let us assume that $B_{x,s,\alpha}$ are all normalized to have unit ℓ_2 norm).

2.1.2 Model

The inhomogeneous FRAME model is a probability distribution defined on $\mathbf{I}$,

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}(\langle \mathbf{I}, B_{x,s,\alpha} \rangle) \right) q(\mathbf{I}), \tag{2.1}$$

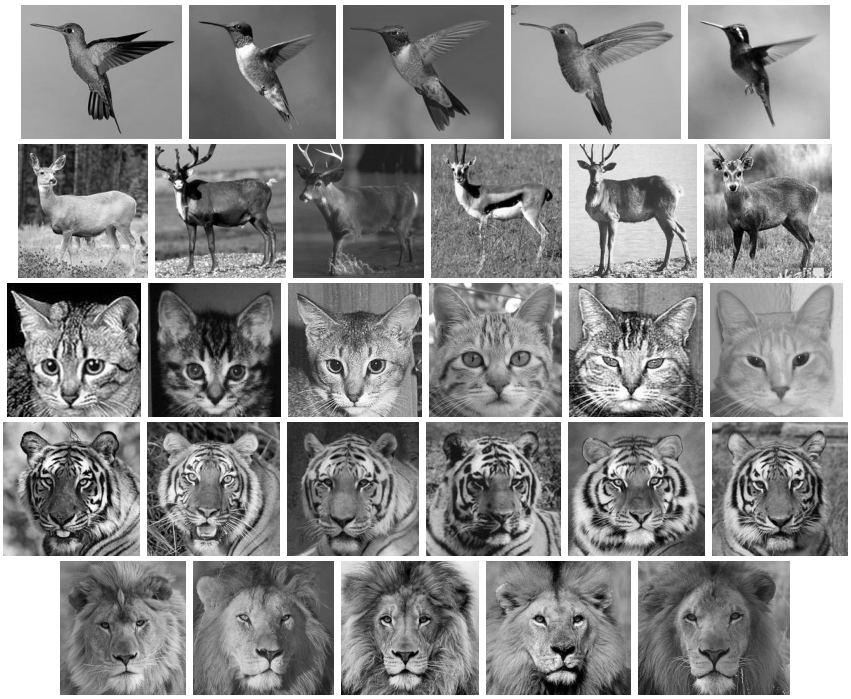

Figure 2.1: The inhomogeneous FRAME is a generative model that seeks to represent and generate object patterns shown above. (From top to bottom: hummingbird, deer, cat, tiger, and lion).

where $q(\mathbf{I})$ is a known reference distribution or a null model, $\lambda_{x,s,\alpha}()$ are one-dimensional functions that depend on (x, s, α), $\lambda = \{\lambda_{x,s,\alpha}, \forall x, s, \alpha\}$, and

$$Z(\lambda) = \int \exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}(\langle \mathbf{I}, B_{x,s,\alpha}\rangle)\right) q(\mathbf{I}) d\mathbf{I} \tag{2.2}$$

$$= \mathrm{E}_q\left[\exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}(\langle \mathbf{I}, B_{x,s,\alpha}\rangle)\right)\right] \tag{2.3}$$

is the normalizing constant, where the notation E_q means the expectation with respect to the probability distribution q.

In the original FRAME model for stochastic textures [23], $\lambda_{x,s,\alpha}()$ is assumed to be independent of x (but dependent of s and α, which index the shapes of the filters), so the model is spatially stationary. For modeling object patterns that are not spatially stationary, $\lambda_{x,s,\alpha}()$ must depend on x, in addition to s and α.

In the original homogeneous FRAME, the potential functions $\lambda_{s,\alpha}()$ (we drop the subscript x due to stationarity) are estimated non-parametrically as step functions. In the inhomogeneous FRAME, we have to estimate $\lambda_{x,s,\alpha}()$ for each individual x. With small data sets, we may not afford estimating $\lambda_{x,s,\alpha}()$ non-parametrically. We therefore decide to parametrize

$$\lambda_{x,s,\alpha}(r) = \lambda_{x,s,\alpha}|r|, \tag{2.4}$$

where $r = \langle \mathbf{I}, B_{x,s,\alpha} \rangle$, and with slight abuse of notation, $\lambda_{x,s,\alpha}$ on the right hand side of the above equation becomes a constant (instead of a function as on the left hand side). The parametrization (2.4) is inspired by the Laplacian distribution that can account for heavy tails in the distributions of filter responses. It is possible to replace the function $|r|$ by other class of parametrized functions to encourage heavy tails of the responses, and we shall investigate this issue in future work.

In many Markov random field models including the original FRAME model, the reference measure $q(\mathbf{I})$ is simply the uniform measure. In our work, we assume $q(\mathbf{I})$ to be the Gaussian white noise model, under which the image intensities follow independent $\mathrm{N}(0, \sigma^2)$ distributions. So

$$q(\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{|\mathcal{D}|/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_x \mathbf{I}(x)^2\right), \tag{2.5}$$

where $|\mathcal{D}|$ is the number of pixels in the image domain. $q(\mathbf{I})$ itself is a maximum entropy model relative to a uniform measure, and it reproduces the marginal mean and variance of the image intensities. In our work, we normalize the observed images to have marginal mean 0 and variance 1, so we choose $\sigma^2 = 1$. This $q(\mathbf{I})$ can be considered an initial model or a model of the background residual image with the foreground object removed. As a result, $p(\mathbf{I}; \lambda)$ in equation (2.1) can be written as an exponential family model relative to a uniform measure.

2.1.3 Maximum likelihood learning

The inhomogeneous version of the FRAME model is a special case of the exponential family model, and the parameter $\lambda = (\lambda_{x,s,\alpha}, \forall x, s, \alpha)$ can be estimated from the training images $\{\mathbf{I}_m, m = 1, ..., M\}$ by maximum likelihood. The log-likelihood function is

$$\begin{aligned} L(\lambda) &= \frac{1}{M}\sum_{m=1}^{M} \log p(\mathbf{I}_m; \lambda) \\ &= \frac{1}{M}\sum_{m=1}^{M}\sum_{x,s,\alpha} \lambda_{x,s,\alpha} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \log Z(\lambda) + \frac{1}{M}\sum_{m=1}^{M} \log q(\mathbf{I}_m), \quad (2.6) \end{aligned}$$

where $Z(\lambda) = \int \exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha} |\langle \mathbf{I}, B_{x,s,\alpha}\rangle|\right) q(\mathbf{I}) d\mathbf{I}$.

The maximization of $L(\lambda)$ can be accomplished by gradient ascent, where the gradient is

$$\begin{aligned} &\frac{\partial L(\lambda)}{\partial \lambda_{x,s,\alpha}} \\ =\ & \frac{1}{M}\sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \frac{1}{Z(\lambda)}\frac{\partial Z(\lambda)}{\partial \lambda_{x,s,\alpha}} \\ =\ & \frac{1}{M}\sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \int \frac{1}{Z(\lambda)} \exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha} |\langle \mathbf{I}, B_{x,s,\alpha}\rangle|\right) q(\mathbf{I}) |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| d\mathbf{I} \\ =\ & \frac{1}{M}\sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \int p(\mathbf{I}; \lambda) |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| d\mathbf{I} \\ =\ & \frac{1}{M}\sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|], \forall x, s, \alpha, \quad (2.7) \end{aligned}$$

where $\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$ is the expectation of $|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|$ with $\mathbf{I}$ following the distribution $p(\mathbf{I}; \lambda)$. $\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$ is the derivative of $\log Z(\lambda)$.

The gradient ascent algorithm then becomes

$$\lambda_{x,s,\alpha}^{(t+1)} = \lambda_{x,s,\alpha}^{(t)} + \gamma_t \left(\frac{1}{M}\sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \mathrm{E}_{p(\mathbf{I};\lambda^{(t)})}\left[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|\right]\right), \quad (2.8)$$

where γ_t is the step size. The analytic form of the expectation under the current model at step t, $\mathrm{E}_{p(\mathbf{I};\lambda^{(t)})}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$, is not available, so we approximate it from a sample set

of synthesized images $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ generated from $p(\mathbf{I}; \lambda^{(t)})$:

$$\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|] \approx \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha}\rangle|. \tag{2.9}$$

The synthesized images $\{\tilde{\mathbf{I}}_m\}$ can be sampled from $p(\mathbf{I}; \lambda^{(t)})$ by the Hamiltonian Monte Carlo (HMC) [13]. More details about the HMC algorithm will be presented in Section 2.2. The computation of HMC involves a bottom-up convolution step followed by a top-down convolution step. Both steps can be efficiently implemented in Matlab by GPU. With HMC and warm start, $\{\tilde{\mathbf{I}}_m\}$ are produced by $\tilde{M}$ parallel chains.

With $\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$ approximated according to (2.9), we arrive at the stochastic gradient algorithm analyzed by Younes (1999) [22] :

$$\lambda_{x,s,\alpha}^{(t+1)} = \lambda_{x,s,\alpha}^{(t)} + \gamma_t \left(\frac{1}{M} \sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha}\rangle| \right). \tag{2.10}$$

This is the algorithm we use for maximum likelihood estimation of λ.

2.1.4 Computing normalizing constants

Thanks to HMC, we can simulate from $p(\mathbf{I}; \lambda)$ without knowing its normalizing constant, thus estimating λ by MLE. Nevertheless, computing normalizing constant $Z(\lambda)$ is still required in situations such as fitting a mixture model or learning a codebook of models. The ratio of the normalizing constants at two consecutive steps is

$$\begin{aligned}
&\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})} \\
= &\int \frac{1}{Z(\lambda^{(t)})} \exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}^{(t)} |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| \right) q(\mathbf{I}) \times \frac{\exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}^{(t+1)} |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| \right)}{\exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}^{(t)} |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| \right)} d\mathbf{I} \\
= &\int p(\mathbf{I}; \lambda^{(t)}) \exp\Big(\sum_{x,s,\alpha} (\lambda_{x,s,\alpha}^{(t+1)} - \lambda_{x,s,\alpha}^{(t)}) \times |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| \Big) d\mathbf{I} \\
= &\mathrm{E}_{p(\mathbf{I};\lambda^{(t)})} \left[\exp\Big(\sum_{x,s,\alpha} (\lambda_{x,s,\alpha}^{(t+1)} - \lambda_{x,s,\alpha}^{(t)}) \times |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| \Big) \right]
\end{aligned} \tag{2.11}$$

which can be approximated by averaging over the sampled images $\{\tilde{\mathbf{I}}_m\}$ as an application of importance sampling [5]:

$$\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})} \approx \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} \left[\exp\Big(\sum_{x,s,\alpha} (\lambda^{(t+1)}_{x,s,\alpha} - \lambda^{(t)}_{x,s,\alpha}) \times |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha} \rangle| \Big) \right]. \tag{2.12}$$

Starting from $\lambda^{(0)} = 0$ and $\log Z(\lambda^{(0)}) = 0$, we can compute $\log Z(\lambda^{(t)})$ along the learning process by iteratively updating its value as follows:

$$\log Z(\lambda^{(t+1)}) = \log Z(\lambda^{(t)}) + \log \frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}. \tag{2.13}$$

The calculation of Z is based on running parallel Markov chains for a sequence of distributions $p(\mathbf{I}; \lambda^{(t)})$. The setting is similar to annealed importance sampling [12] and bridge sampling [5].

2.2 Simulation by Hamiltonian Monte Carlo

To approximate $\mathrm{E}_{p(\mathbf{I};\lambda^{(t)})}[|\langle \mathbf{I}, B_{x,s,\alpha} \rangle|]$ in equation (2.8), we need to draw a synthesized sample set $\{\tilde{\mathbf{I}}_m\}$ from $p(\mathbf{I}; \lambda^{(t)})$ by HMC [4]. HMC is a Markov chain Monte Carlo method using Hamiltonian dynamics. It requires translating the probability distribution we wish to sample from to a potential energy function and introducing auxiliary momentum variables, which typically follow independent Gaussian distributions, to go with the original variables we are interested in. Then a Markov chain can be simulated by two steps: (1) generating the momentum by sampling from Gaussian distribution, and (2) performing a Metropolis update with a proposal found by Hamiltonian dynamic. Unlike Gibbs sampler [6], HMC avoids diffusive random walk behavior, which might lead to the slow exploration of the state space, by simulating the evolution of a physical system under Hamiltonian dynamics [13]. Moreover, when sampling from continuous variables such as $\mathbf{I}$, HMC can prove to be a powerful and efficient tool [4]. Also, HMC makes use of the gradient of the energy function, and it is particularly natural for the inhomogeneous FRAME model.

To draw samples from inhomogeneous FRAME model $p(\mathbf{I};\lambda)$, we can write $p(\mathbf{I};\lambda)$ as $p(\mathbf{I}) \propto \exp(-U(\mathbf{I}))$, where $\mathbf{I} \in R^{|\mathcal{D}|}$ and $U(\mathbf{I}) = -\sum_{x,s,\alpha} \lambda_{x,s,\alpha} |\langle \mathbf{I}, B_{x,s,\alpha} \rangle| + \frac{1}{2}|\mathbf{I}|^2$ (assuming $\sigma^2 = 1$). In physics context, $\mathbf{I}$ can be regarded as a position vector and $U(\mathbf{I})$ the potential energy function. To allow Hamiltonian dynamics to operate, we need to introduce an auxiliary momentum vector $\boldsymbol{\phi} \in R^{|\mathcal{D}|}$ and the corresponding kinetic energy function $K(\boldsymbol{\phi}) = |\boldsymbol{\phi}|^2/2m$, where m represents the mass. After that, a fictitious physical system described by the canonical coordinates $(\mathbf{I}, \boldsymbol{\phi})$ is defined, and its total energy is $H(\mathbf{I}, \boldsymbol{\phi}) = U(\mathbf{I}) + K(\boldsymbol{\phi})$. Instead of sampling from $p(\mathbf{I})$ directly, HMC samples from the joint canonical distribution $p(\mathbf{I}, \boldsymbol{\phi}) \propto \exp(-H(\mathbf{I}, \boldsymbol{\phi}))$ by simulating the evolution of the physical system using Hamiltonian dynamics, however, because $\mathbf{I}$ and $\boldsymbol{\phi}$ are independent, marginalizing over $\boldsymbol{\phi}$ to retrieve the original target distribution $p(\mathbf{I})$ is trivial. If a physical system evolves under Hamiltonian dynamics, it will conserve the total energy (i.e. $H(\mathbf{I}, \boldsymbol{\phi})$ remains as a constant). Therefore, such a dynamics can be used as transition operators of a Markov chain and will lead to invariant $p(\mathbf{I}, \boldsymbol{\phi})$ in MCMC sampling. As to our case, the time evolution under Hamiltonian dynamics can be defined by Hamilton's equations [7] as follows

$$\frac{d\mathbf{I}}{dt} = \frac{\partial H}{\partial \boldsymbol{\phi}} = \frac{\boldsymbol{\phi}}{m}, \tag{2.14}$$

$$\frac{d\boldsymbol{\phi}}{dt} = -\frac{\partial H}{\partial \mathbf{I}} = -\frac{\partial U}{\partial \mathbf{I}}. \tag{2.15}$$

In practical implementation, the leapfrog algorithm is used to discretize the continuous Hamiltonian dynamics as follows, with ϵ being the step-size:

$$\boldsymbol{\phi}^{(t+\epsilon/2)} = \boldsymbol{\phi}^{(t)} - (\epsilon/2)\frac{\partial U}{\partial \mathbf{I}}(\mathbf{I}^{(t)}), \tag{2.16}$$

$$\mathbf{I}^{(t+\epsilon)} = \mathbf{I}^{(t)} + \epsilon\frac{\boldsymbol{\phi}^{(t+\epsilon/2)}}{m}, \tag{2.17}$$

$$\boldsymbol{\phi}^{(t+\epsilon)} = \boldsymbol{\phi}^{(t+\epsilon/2)} - (\epsilon/2)\frac{\partial U}{\partial \mathbf{I}}(\mathbf{I}^{(t+\epsilon)}), \tag{2.18}$$

that is, a half-step update of $\boldsymbol{\phi}$ is performed first and then it is used to compute $\mathbf{I}^{(t+\epsilon)}$ and $\boldsymbol{\phi}^{(t+\epsilon)}$.

A key step in the leapfrog algorithm is the computation of the derivative of the potential energy function

$$\frac{\partial U}{\partial \mathbf{I}} = -\sum_{x,s,\alpha} \lambda_{x,s,\alpha} \text{sign}(\langle \mathbf{I}, B_{x,s,\alpha} \rangle) B_{x,s,\alpha} + \mathbf{I}, \tag{2.19}$$

where the map of responses $r_{x,s,\alpha} = \langle \mathbf{I}, B_{x,s,\alpha} \rangle$ is computed by bottom-up convolution of the filter corresponding to (s, α) with $\mathbf{I}$ for each (s, α). Then the derivative is computed by top-down linear superposition of the basis functions: $-\sum_{x,s,\alpha} \lambda_{x,s,\alpha} \text{sign}(r_{x,s,\alpha}) B_{x,s,\alpha} + \mathbf{I}$, which can again be computed by convolution. Both bottom-up and top-down convolutions can be carried out efficiently by GPUs.

The discretization of the leapfrog algorithm cannot keep $H(\mathbf{I}, \boldsymbol{\phi})$ exactly constant, so a Metropolis acceptance/rejection step is used to correct the discretization error. Starting with the current state, $(\mathbf{I}, \boldsymbol{\phi})$, the new state $(\mathbf{I}^\star, \boldsymbol{\phi}^\star)$ after L leapfrog steps is accepted as the next state of the Markov chain with probability

$$\min \left(1, \exp(-H(\mathbf{I}^\star, \boldsymbol{\phi}^\star) + H(\mathbf{I}, \boldsymbol{\phi})) \right). \tag{2.20}$$

If it is not accepted, the next state is the same as the current state.

In summary, a complete description of the HMC sampler for inhomogeneous FRAME is as follows:

(i) Generate momentum vector $\boldsymbol{\phi}$ from $p(\boldsymbol{\phi}) \propto \exp(-K(\boldsymbol{\phi}))$, which is the zero-mean Gaussian distribution with covariance matrix mI. (I is identity matrix).

(ii) Perform L leapfrog steps to reach the new state $(\mathbf{I}^\star, \boldsymbol{\phi}^\star)$.

(iii) Perform acceptance/rejection of the proposed state $(\mathbf{I}^\star, \boldsymbol{\phi}^\star)$.

L, ϵ, and m are parameters of the algorithm, which need to be tuned to obtain good performance.

All the synthesized images presented in the figures of this paper are generated by the HMC algorithm along the learning process.

2.3 Summary of the learning algorithm

Pseudocode of the algorithm for learning the inhomogeneous FRAME model is shown in Algorithm 1. The algorithm stops when the gradient of the log-likelihood is close to 0, i.e., when the statistics of the synthesized images closely match those of the observed images. Figure 2.2 displays the synthesized images $\{\tilde{\mathbf{I}}_m\}$ generated by the models learned from training images shown in Figure 2.1 (a separate model is learned from each training set). Figure 2.3 illustrates the learning process by showing the synthesized images with λ being updated by the algorithm. The synthesized image starts from Gaussian white noises sampled from $q(\mathbf{I})$, then gradually gets similar to the observed images in the overall shape and appearance.

The computational complexity of the Algorithm 1 is $O(U \times \tilde{M} \times L \times K \times H_B \times W_B)$ with U the number of updating steps for λ, $\tilde{M}$ the number of synthesized images, L the number of leapfrog steps in HMC, K the number of filters, and H_B and W_B are the average window sizes (height and width) of the filters. As to the actual running time, for the cat example, each iteration of a single chain takes about 2 seconds on a current PC, with $L = 30$, $K = 240100$, $H_B = 12$, and $W_B = 12$.

2.4 Experiment 1: Synthesis by inhomogeneous FRAME model

Figure 2.4 displays some images generated by the inhomogeneous FRAME models learned from roughly aligned training images. We run a single chain in the learning process, i.e., $\tilde{M} = 1$ in this experiment. The learned models can generate a wide variety of natural image patterns. Typical sizes of the images are 70×70. We use a filter bank containing Gabor filters at 3 scales and DoG filters at one scale. We run 500 iterations.

Algorithm 1 The learning algorithm for inhomogeneous FRAME

Input:

training images $\{\mathbf{I}_m, m = 1, ..., M\}$

Output:

$\lambda = \{\lambda_{x,s,\alpha}, \forall x, s, \alpha\}$ and $\log Z(\lambda)$

1: Create a filters bank $\{B_{x,s,\alpha}, \forall x, s, \alpha\}$

2: Initialize $\lambda^{(0)}_{x,s,\alpha} \leftarrow 0, \forall x, s, \alpha$.

3: Calculate observed statistics:

$$H^{obs}_{x,s,\alpha} \leftarrow \frac{1}{M} \sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha} \rangle|, \forall x, s, \alpha.$$

4: Initialize synthesized images $\tilde{\mathbf{I}}_m$ as Gaussian white noises images

5: Initialize $\log Z(\lambda^{(0)}) \leftarrow 0$

6: Let $t \leftarrow 0$

7: **repeat**

8: Generate $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ from $p(\mathbf{I}; \lambda^{(t)})$ by HMC

9: Calculate synthesized statistics:

$$H^{syn}_{x,s,\alpha} \leftarrow \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha} \rangle|, \forall x, s, \alpha.$$

10: Update $\lambda^{(t+1)}_{x,s,\alpha} \leftarrow \lambda^{(t)}_{x,s,\alpha} + \gamma_t (H^{obs}_{x,s,\alpha} - H^{syn}_{x,s,\alpha}), \forall x, s, \alpha$.

11: Compute Z ratio $\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}$ by Eq. (2.12)

12: Update $\log Z(\lambda^{(t+1)}) \leftarrow \log Z(\lambda^{(t)}) + \log \frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}$

13: Let $t \leftarrow t + 1$

14: **until** $\sum_{x,s,\alpha} |H^{obs}_{x,s,\alpha} - H^{syn}_{x,s,\alpha}| \leq \epsilon$

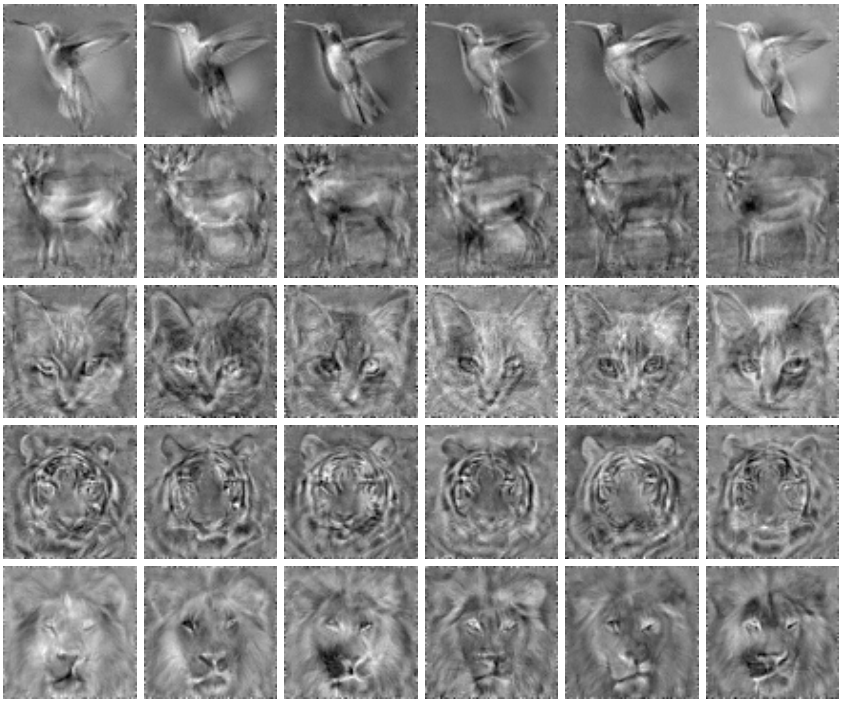

Figure 2.2: Synthesized images generated by the inhomogeneous FRAME model. The sizes of the images are 70×70. Each row contains 6 independent samples (synthesized images) drawn from the learned model.

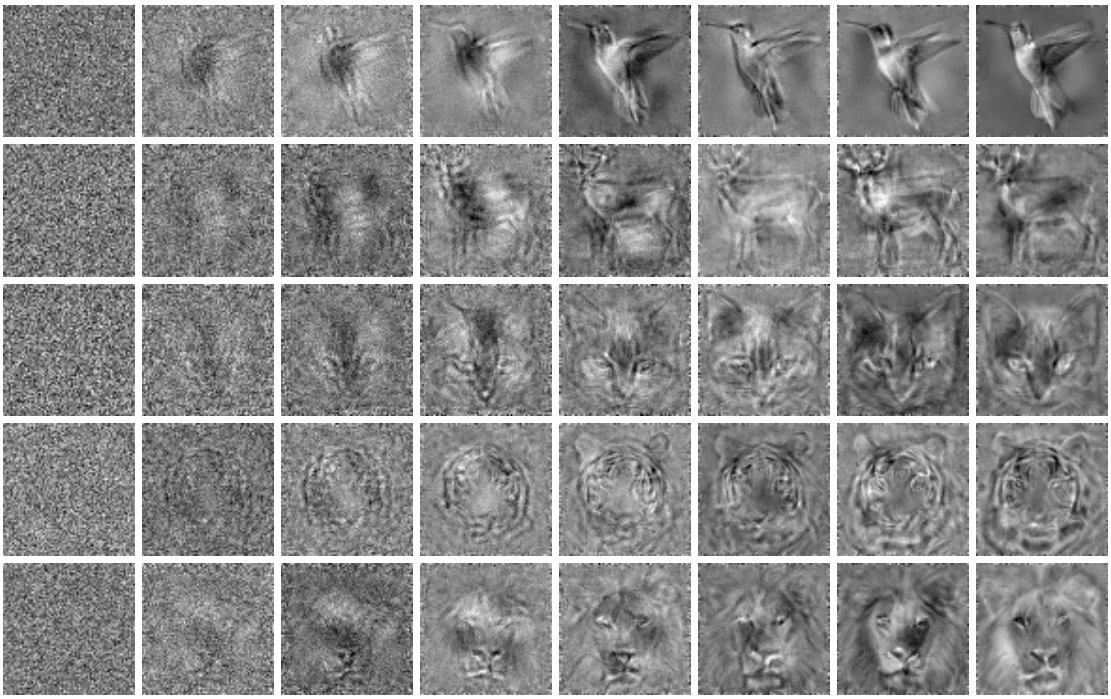

Figure 2.3: Learning sequence by inhomogeneous FRAME. The size of the images are 70×70. A separate model is learned from each training set shown in Fig 1 (From top to bottom: hummingbird, deer, cat, tiger, and lion). Synthesized images generated in step $t = 1, 7, 10, 20, 50, 100, 200$, and 500.

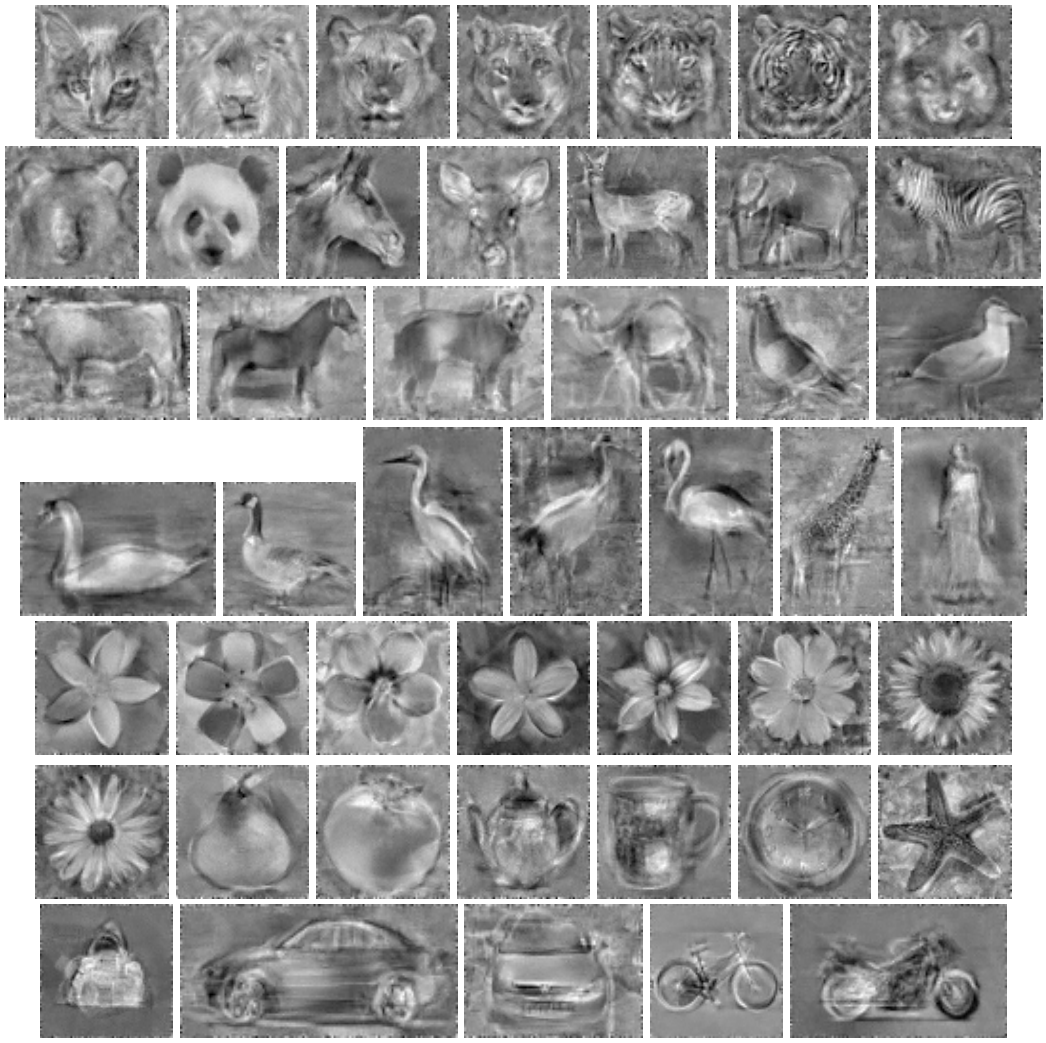

Figure 2.4: Images generated by the inhomogeneous FRAME model learned from different categories of objects. The training images are collected from the internet and are cropped so that the training images for each category are roughly aligned. The number of training images for each category is around 10.

CHAPTER 3

Detection

3.1 Detection algorithm

After learning the inhomogeneous FRAME model $p(\mathbf{I}; \lambda)$, where $\lambda = (\lambda_{x,s,\alpha}, \forall x, s, \alpha)$, from the roughly aligned training images $\{\mathbf{I}_m, m = 1, ..., M\}$, the learned model $p(\mathbf{I}; \lambda)$ can serve as a template, so that we can use it to detect the object in a testing image by template matching.

Let $\mathbf{I}$ be a testing image defined on the domain $\mathcal{D}$. We can scan the template over $\mathcal{D}$, and at each location $X \in \mathcal{D}$, we match the template to the image patch of $\mathbf{I}$ within the bounding box centered at X by computing the log-likelihood

$$L(\mathbf{I} \mid \lambda) = \sum_{x,s,\alpha} \lambda_{x,s,\alpha} |\langle \mathbf{I}, B_{x,s,\alpha} \rangle| - \log Z(\lambda), \tag{3.1}$$

which serves as template matching score. We then choose the location X that achieves the maximum template matching score as the center of the detected object. To deal with the scaling issue, we can apply the above algorithm at multiple resolutions of the testing image, and then choose the resolution that achieves the maximum template matching score as the optimal resolution.

In addition to spatial translation in scanning, we can also allow geometric transformations such as rotation and left-right flipping of the template. The geometrically transformed versions of the learned model can be obtained by directly rotating or flipping the learned parameter map $\lambda = \{\lambda_{x,s,\alpha}, \forall x, s, \alpha\}$ without recomputing the values of λ. This amounts to simple affine transformations of $(x, s, \alpha, \forall x, s, \alpha)$. For better

performance in detection, we can first generate a collection of models at different orientations with and without left/right flipping from the learned model. After that, we use these transformed models to detect the object. We choose the combination of the transformed template and image resolution that gives the best match in terms of the template matching score, and infer the hidden location, orientation, and scale of the detected object in the testing image.

3.2 Experiment 2: Detection by template matching

Experiment 2a: Figure 3.1 shows examples of detection in tiger category. We learn the model from six roughly aligned training images, with $\tilde{M} = 36$. We don't use DoG filters in learning for detection. The template size is 100×100. The image displayed on the top row is a synthesized image generated by the learned model. We transform the learned model into a collection of models at 9 different orientations without left/right flipping, and then run the detection algorithm over 9 resolutions of the testing images using these transformed templates. The detection results are displayed by drawing bounding boxes on the detected objects.

Experiment 2b: Figure 3.2 shows another case of detection in wolf category, where the model is learned from 16 training images, the detection algorithm is run over 7 resolutions. Other tuning parameters are the same as those in the last case.

Experiment 2c: Figure 3.3 shows the third case of detection in zebra category. In this case, the number of training images is 7. Template size is 117×150. Seven different orientations as well as left/right flipping are considered in transforming the learned template for detection. A small square attached to the top left corner of the bounding box is for distinguishing detected objects with or without left/right flipping.

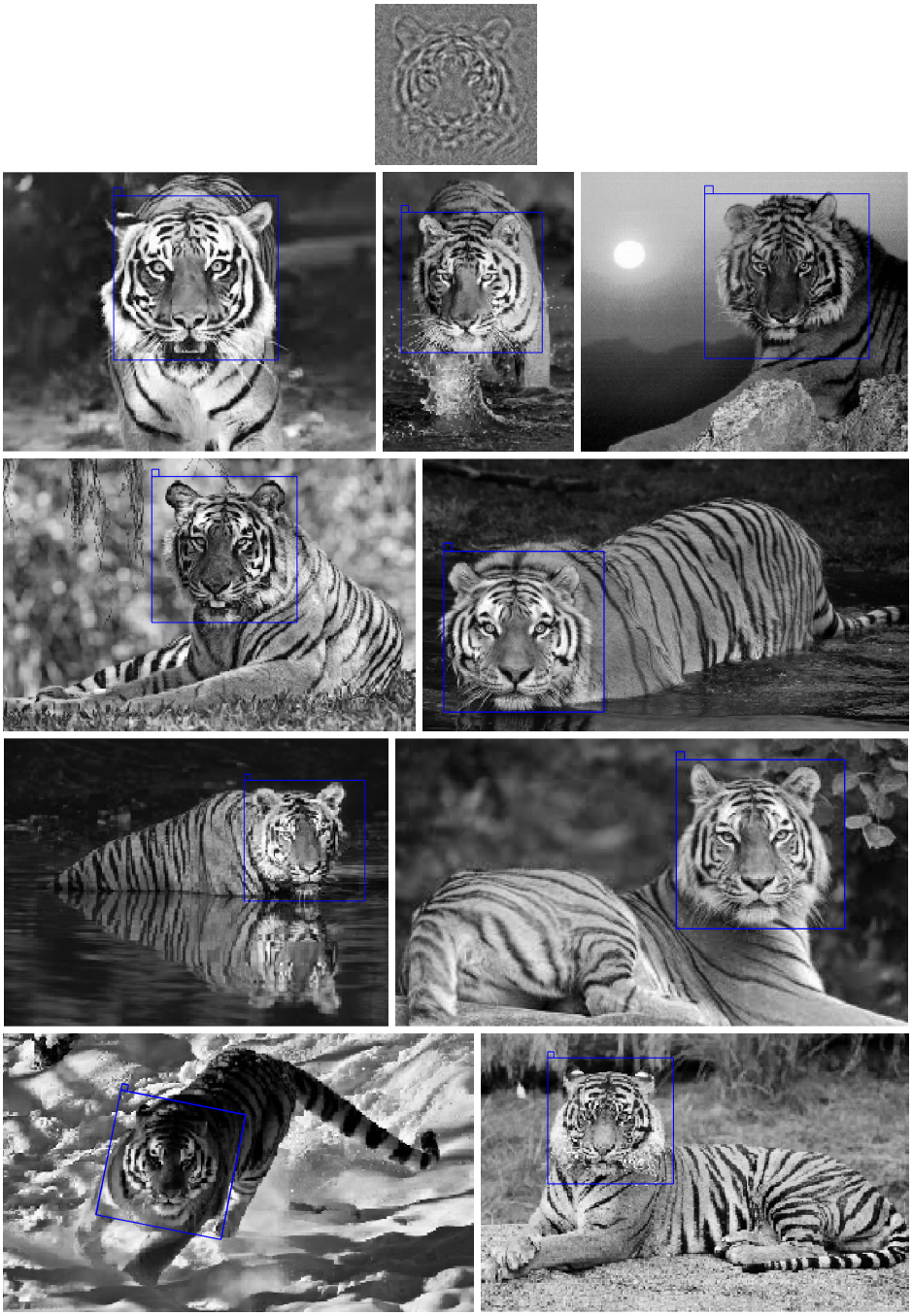

Figure 3.1: Detection (tigers). On the top: A synthesized image generated by the learned model. The rest: Testing images with bounding boxes locating the detected objects.

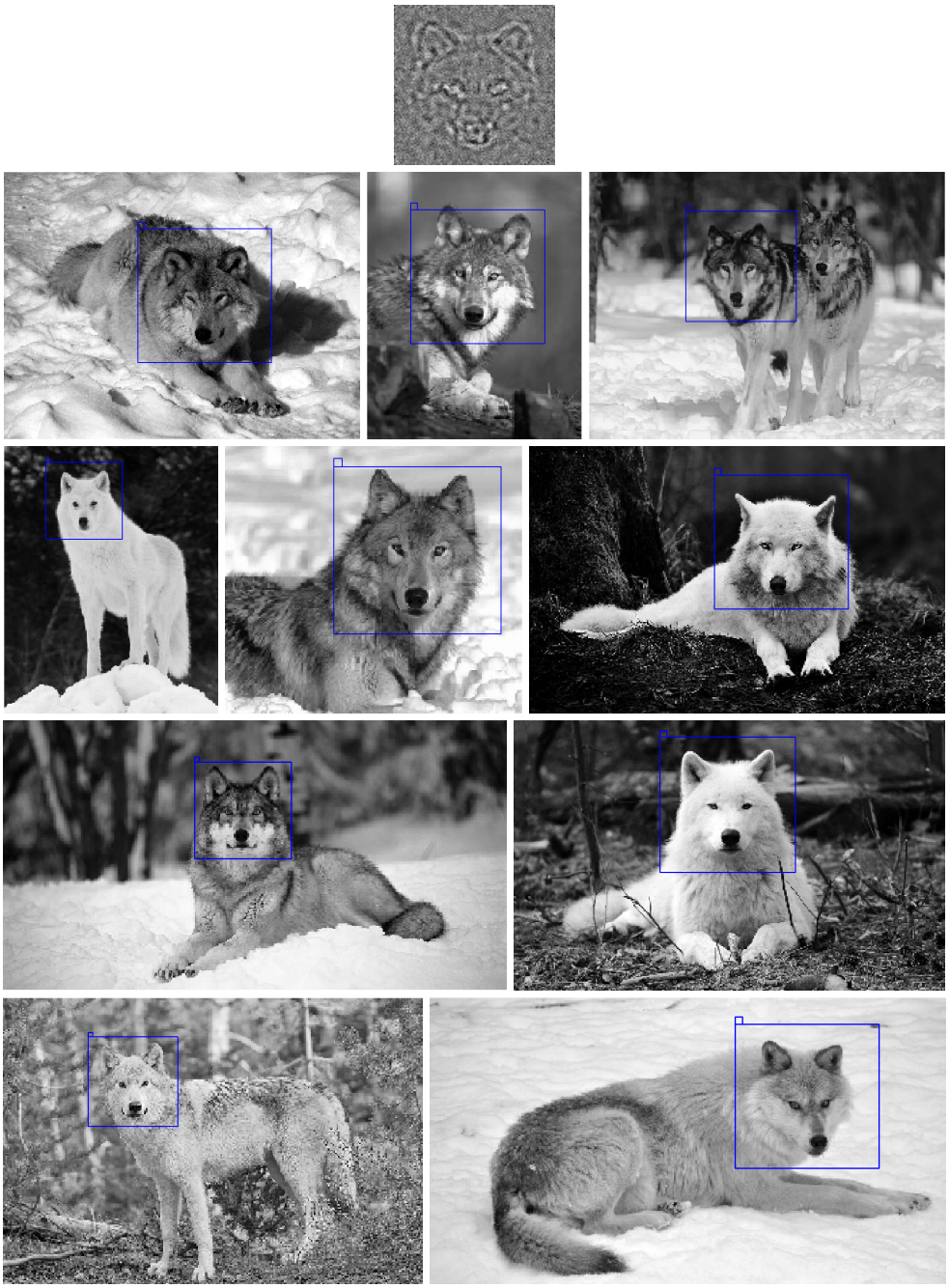

Figure 3.2: Detection (wolves). On the top: A synthesized image generated by the learned model. The rest: Testing images with bounding boxes locating the detected objects.

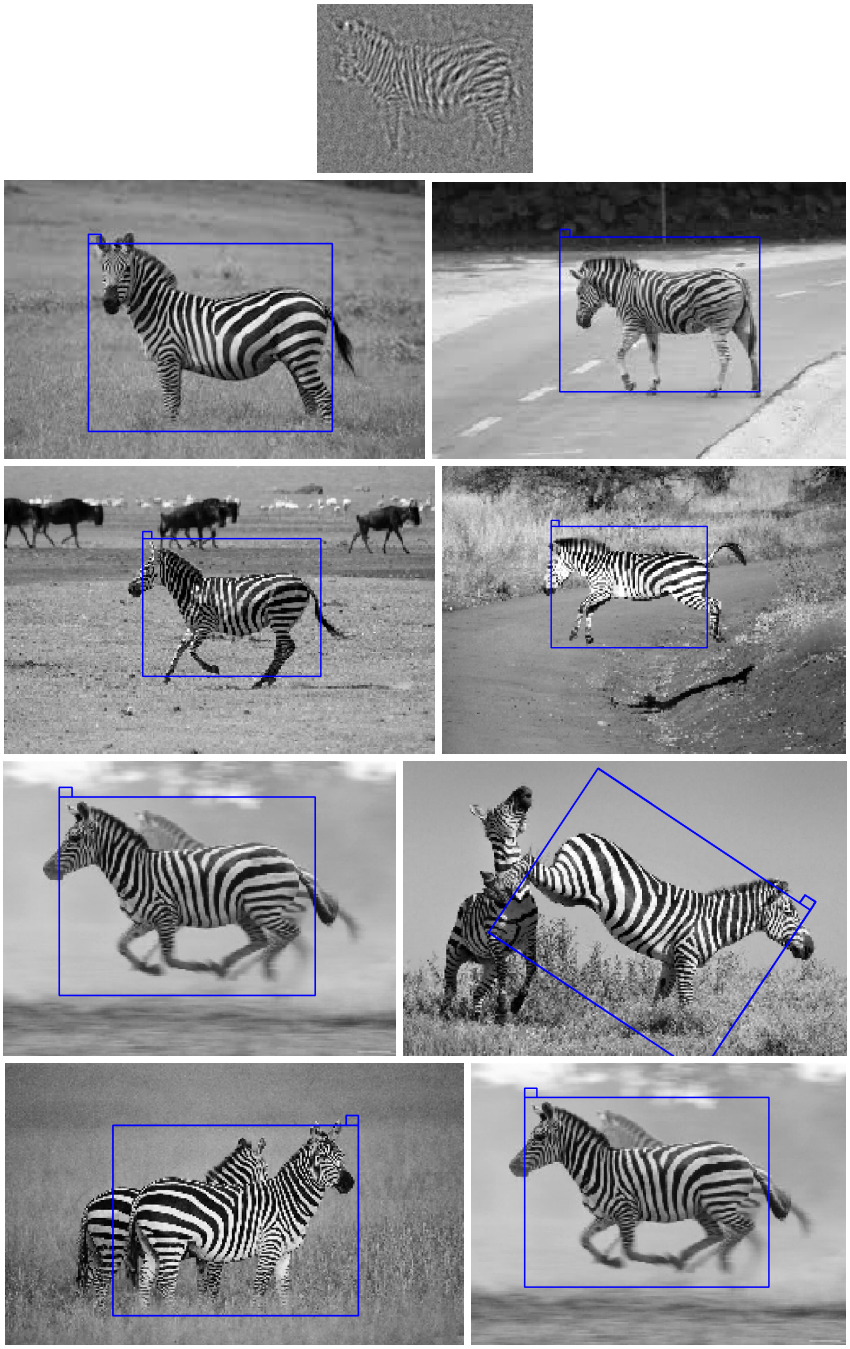

Figure 3.3: Detection (zebras). On the top: A synthesized image generated by the learned model. The rest: Testing images with bounding boxes locating the detected objects.

CHAPTER 4

Alignment

4.1 Alignment algorithm

In the previous section, training images $\{\mathbf{I}_m, m = 1, ..., M\}$ are defined on the same bounding box. In this section, we study the problem of learning from images where the objects are of unknown locations, orientations and scales. It is also called multiple images alignment, which is an unsupervised learning problem. We start from the multiple alignment score defined by

$$\text{ALIGN}(\mathbf{I}_m, m = 1, ..., M) = \sum_{m=1}^{M} L(\mathbf{I}_m|\lambda), \tag{4.1}$$

where $L(\mathbf{I}_m|\lambda)$ is the log-likelihood or template matching score defined by equation (3.1). However, when the training images $\{\mathbf{I}_m, m = 1, ..., M\}$ are of different sizes, and the objects appear at different locations, orientations, and scales in the training images, we need to infer the unknown locations, orientations, and scales. We denote box(x, s, α) as the rectangular bounding box of the template centered at location x, with orientation α and scale s. For an image $\mathbf{I}$, let $\mathbf{I}[\text{box}(x, s, \alpha)]$ be the image patch cropped from the image $\mathbf{I}$ within box(x, s, α). Our goal is to maximize the alignment score

$$\text{ALIGN}(\mathbf{I}_m[\text{box}(x_m, s_m, \alpha_m)], m = 1, ..., M) \tag{4.2}$$

over $\{(x_m, s_m, \alpha_m), m = 1, ..., M\}$, where (x_m, s_m, α_m) is the unknown location, orientation, scale of the bounding box in $\mathbf{I}_m$. A greedy strategy that iterates the following two steps can be used to maximize equation (4.2):

(1) *Learning*: Given $\{(x_m, s_m, \alpha_m), m = 1, ..., M\}$, learning $p(\mathbf{I}; \lambda)$ from cropped images $\{\mathbf{I}_m[\text{box}(x_m, s_m, \alpha_m)], m = 1, ..., M\}$ by the learning algorithm for inhomogeneous FRAME described in Algorithm 1.

(2) *Detection*: Given $p(\mathbf{I}; \lambda)$, estimate $\{(x_m, s_m, \alpha_m), m = 1, ..., M\}$ from each $\mathbf{I}_m$ using detection algorithm described in section 3.

4.2 Experiment 3: Multiple images alignment

Experiment 3a: We initialize the algorithm by assuming that the whole image lattice is the initial bounding box of each training image. The template size is 100×100. In step (2), we search over 7 different scales and 3 different orientations. Instead of directly scaling the template, we use 7 different resolutions of the input images (from 0.7 to 1.3 times the input image size). We crop $\mathbf{I}_m[\text{box}(x, \alpha)]$ from the optimum resolution. The algorithm is run for 5 iterations. Figure 4.1 displays one example of alignment results, where bounding boxes are placed on the aligned objects in training images to show the inferred locations, orientations, and scales. Figure 4.2 shows the cropped objects.

Experiment 3b: Different from Experiment 3a, this experiment has one more hidden variable f, the unknown left/right flipping indicator (it takes 1 when flipping happens and 0 otherwise), to infer. Now, we simply replace the notation $\text{box}(x, s, \alpha)$ by $\text{box}(x, s, \alpha, f)$ to represent the rectangular bounding box centered at location x, with orientation α, scale s, and left/right flipping f. Let $\{\mathbf{I}_m[\text{box}(x_m, s_m, \alpha_m, f)], m = 1, ..., M\}$ be the cropped image patches. The alignment leads to maximize the objective function: $\text{ALIGN}(\mathbf{I}_m[\text{box}(x_m, s_m, \alpha_m, f_m)], m = 1, ..., M)$. Figure 4.3 shows one example, where a small square attached to the top left corner of the bounding box is used for distinguishing objects with or without left/right flipping. Figure 4.4 shows the cropped image patches after alignment.

Figure 4.1: Alignment results (deers). Training images with bounding boxes locating the aligned objects.

Figure 4.2: Cropped objects after alignment (deers)

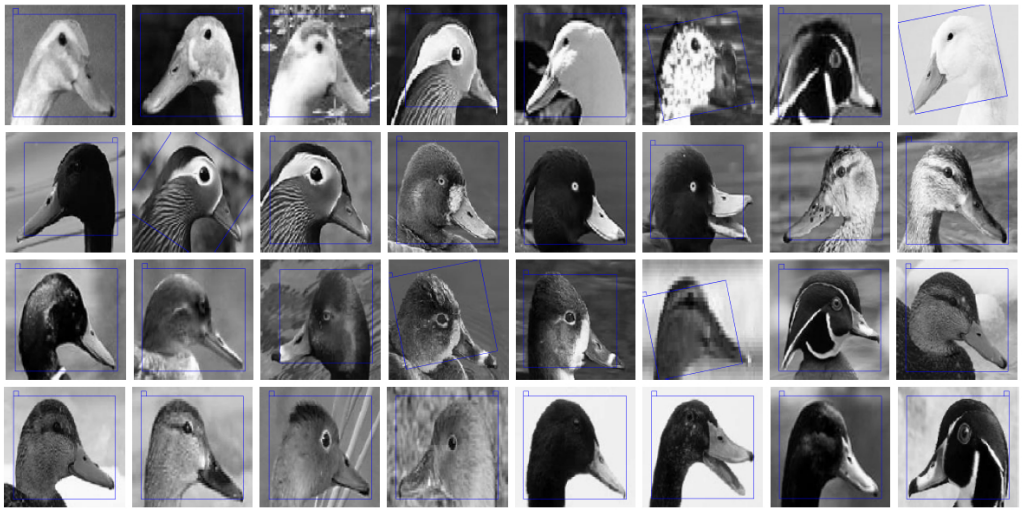

Figure 4.3: Alignment results (ducks). Training images with bounding boxes locating the aligned objects.

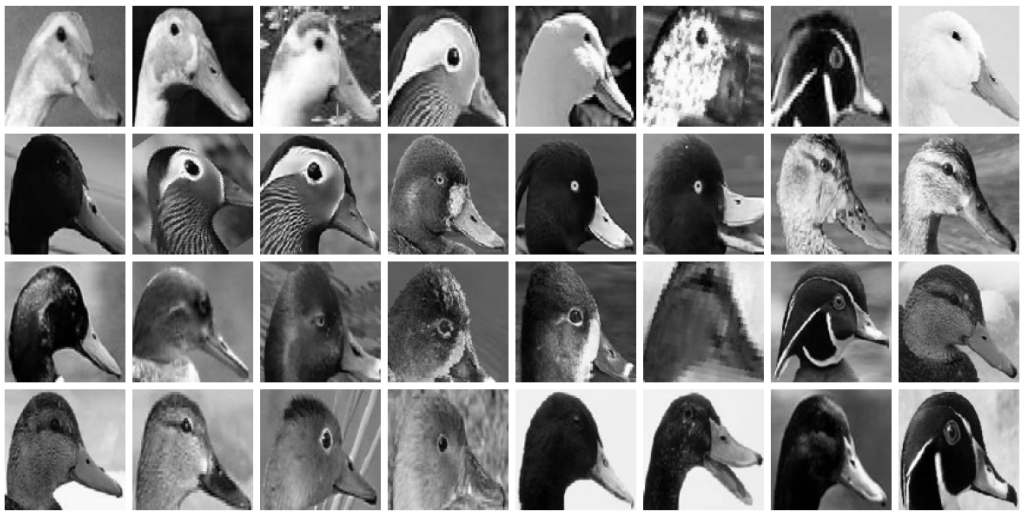

Figure 4.4: Cropped objects after alignment (ducks)

CHAPTER 5

Clustering

In this section, we study the problem of clustering. Unlike conventional clustering problem, we not only need to separate the examples into different clusters, but also need to learn inhomogeneous FRAME model for each cluster. We will discuss two types of clustering algorithms: (1) EM algorithm, and (2) k-mean algorithm.

5.1 Mixture of inhomogeneous FRAME models and EM

Suppose we have M images from K clusters, and each cluster k can be modeled by an inhomogeneous FRAME model $p(\mathbf{I}; \lambda^{(k)})$. A mixture distribution for these images can be described by

$$\sum_{k=1}^{K} \rho^{(k)} p(\mathbf{I}_m; \lambda^{(k)}), \tag{5.1}$$

where $\rho^{(k)}$ is the probability of a training image coming from cluster k, $k = 1, .., K$. For each image $\mathbf{I}_m$, we define $(z_m^{(k)}, k = 1, ..., K)$ as a hidden indicator vector, where $z_m^{(k)} = 1$ if $\mathbf{I}_m$ comes from cluster k, otherwise $z_m^{(k)} = 0$. Model-based clustering can be accomplished by EM algorithm [3] that fits a mixture of inhomogeneous FRAME models. EM starts with randomly generated $\{z_m^{(k)}\}$ and then iterates the following M-step and E-step until convergence.

M-step: For each cluster $k = 1, ..., K$, we learn $p(\mathbf{I}; \lambda^{(k)})$ by the learning algorithm in Algorithm 1. We only need to change the calculation of the observed statistics in

step (3) in the original version of the learning algorithm into

$$H^{obs}_{x,s,\alpha} \leftarrow \frac{\sum_{m=1}^{M} z_m^{(k)} |\langle \mathbf{I}_m, B_{x,s,\alpha} \rangle|}{\sum_{m=1}^{M} z_m^{(k)}}, \forall x, s, \alpha, \tag{5.2}$$

which is a weighted average. Also, for each $k = 1, ..., K$, we need to compute $\rho^{(k)} = \frac{1}{M} \sum_{m=1}^{M} z_m^{(k)}$.

E-step: For each $m = 1, ..., M$ and $k = 1, ..., K$, we compute

$$z_m^{(k)} = \frac{\rho^{(k)} \exp\{L(\mathbf{I}_m|\lambda^{(k)}\})}{\sum_{k=1}^{K} \rho^{(k)} \exp\{L(\mathbf{I}_m|\lambda^{(k)}\})}. \tag{5.3}$$

$z_m^{(k)}$ becomes a fraction due to a soft classification of image $\mathbf{I}_m$ based on current learned model of cluster k.

5.2 Model-based k-mean clustering

In this section, we use the same notations as those used in the previous section for convenience. The model-based k-mean clustering is different from the conventional k-mean algorithm. Instead of using simple average as mean vector, we learn the inhomogeneous FRAME model. Also, the distance is not simple Euclidean distance. Instead, it is measured by template matching score or log-likelihood defined by equation (3.1). The algorithm is a greedy scheme that infers $\{z_m^{(k)}\}$ and $\{\lambda^{(k)}, k = 1, ..., K\}$ by maximizing the overall log-likelihood

$$\sum_{k=1}^{K} \sum_{m=1}^{M} z_m^{(k)} L(\mathbf{I}_m|\lambda^{(k)}), \tag{5.4}$$

where $\lambda^{(k)}$ are the learned parameters for cluster k, and $L(\mathbf{I}_m|\lambda^{(k)})$ is the log-likelihood or template matching score defined by equation (3.1).

The algorithm is initialized by randomly generating $\{z_m^{(k)}\}$, and then iterates the following two steps:

Re-learning: Given $\{(z_m^{(k)}, k = 1, ..., K), m = 1, ..., M\}$, learn the inhomogeneous FRAME model $p(\mathbf{I}; \lambda^{(k)})$ from images classified into the k-th cluster: $\{\mathbf{I}_m, z_m^k = 1\}$, for each $k = 1, ..., K$.

Classification: Given the learned models of the K clusters: $\{p(\mathbf{I}; \lambda^{(k)}), k = 1, ..., K\}$, assign each image $\mathbf{I}_m$ to a cluster k_* that maximizes the template matching score $L(\mathbf{I}_m|\lambda^{(k)})$ over all $k = 1, ..., K$. Set $z_m^{(k_*)} = 1$, and set $z_m^{(k)} = 0$ for $k \neq k_*$

In the above algorithm, the classification step corresponds to the E-step of the EM algorithm, except that we adopt hard classification, instead of computing the expectation of z_m for each image $\mathbf{I}_m$. The re-learning step corresponds to the M-step of the EM algorithm. The algorithm usually converges within a few iterations.

5.3 Experiment 4: Model-based clustering

Experiment 4a: Figure 5.1 illustrates one clustering example using EM algorithm. No DoG filters are used in this experiment. The EM algorithm converges after 4 iterations, at which point all the images are correctly separated into their respective clusters. For each cluster, we generate $\tilde{M} = 144$ parallel chains in learning, because we need to compute $Z(\lambda)$ accurately for each model, as multiple models compete to explain the images. Template sizes are 60×80.

Experiment 4b: Figure 5.2 displays 5 clustering examples using the model-based k-mean algorithm. Each row illustrates one clustering example by displaying a synthesized image and a typical training example for each cluster. The typical number of the training images in each cluster for each example is 15. The k-mean algorithm usually converges within 3-5 iterations. We use DoG filters in this experiment.

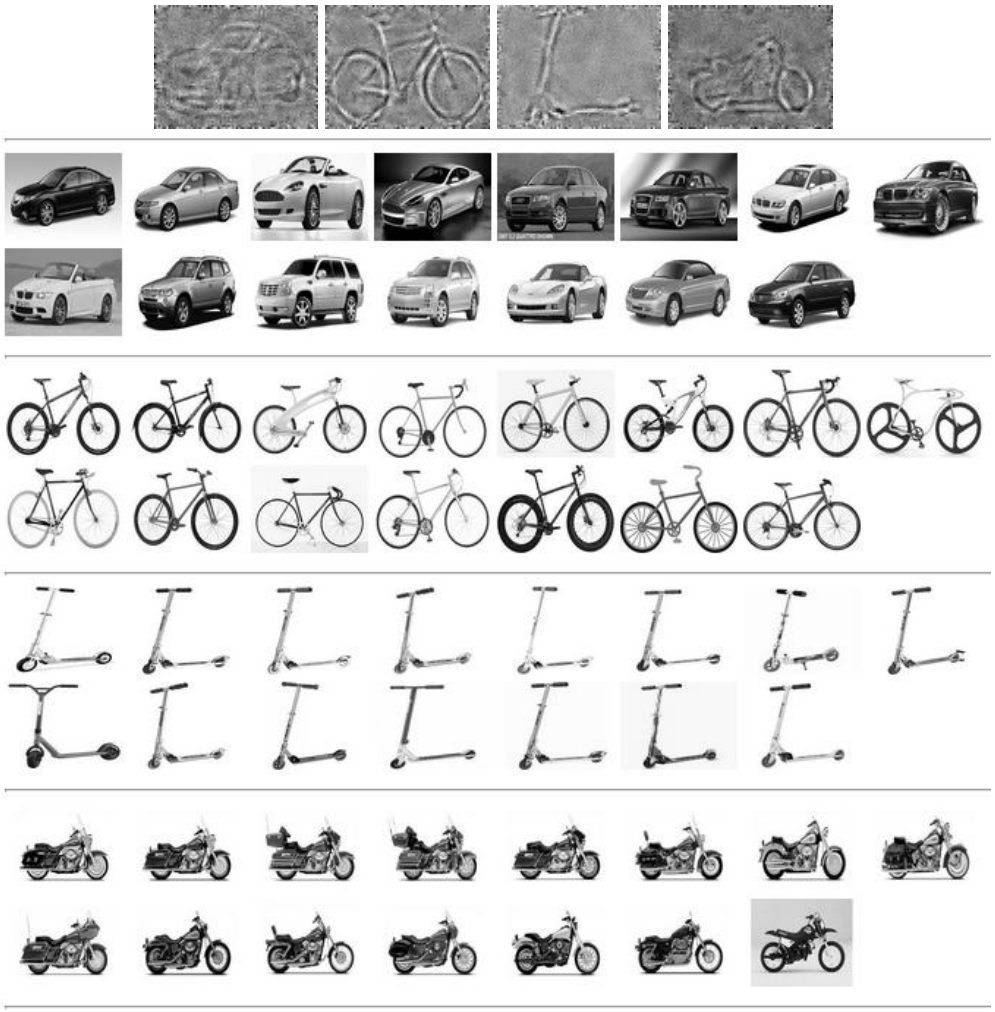

Figure 5.1: EM clustering: On the top: A synthesized image for each cluster is displayed. The rest: 4 clusters of images separated by the model-based EM algorithm.

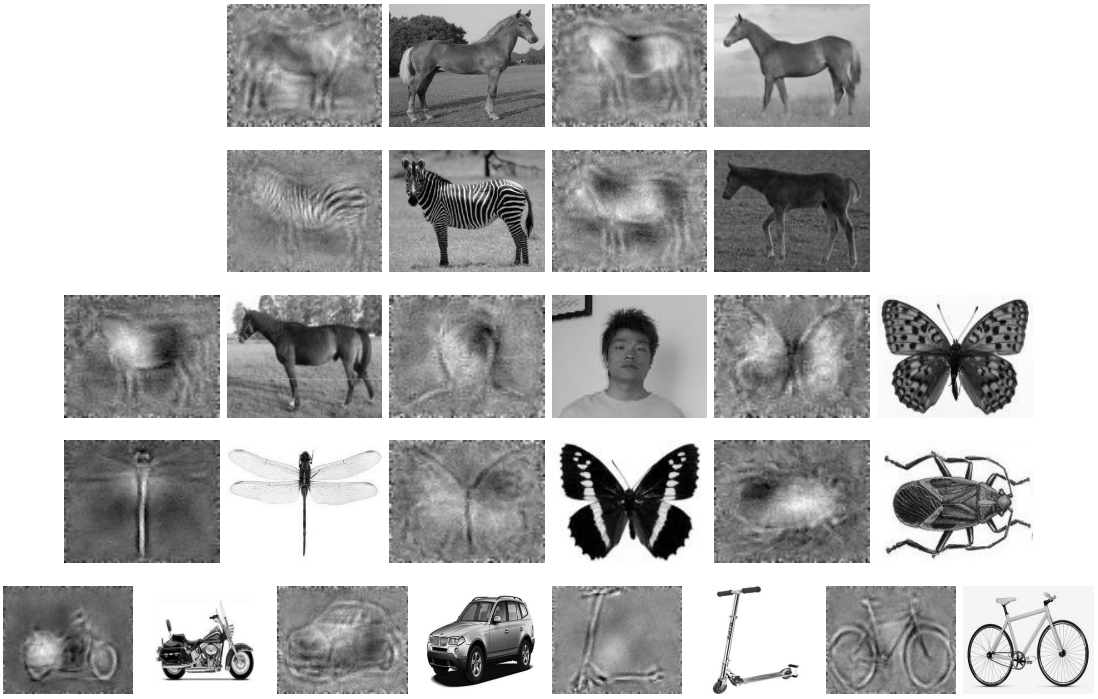

Figure 5.2: k-mean clustering. Each row illustrates one clustering example by displaying a synthesized image and a typical training example for each cluster.

CHAPTER 6

Discussion

Developing generative models for image patterns is one of the most fundamental problems in vision. Such models provide knowledge representations for image patterns, and they may be learned in unsupervised manner so that the learned models can be useful for various vision tasks. Although the past decade has witnessed tremendous advance in developing discriminative methods for object recognition, the progress in developing generative models for object patterns has been lagging behind. In this thesis, a generative model, inhomogeneous FRAME model, for object patterns is proposed under the above motivation.

The proposed model has the following merits: (1) It can be considered a further step in Markov random field or the original FRAME model. (2) It can synthesize new vivid images. In fact, synthesis is required for estimating parameters and calculating the normalizing constant. (3) It is an appearance model that encompasses sketch patterns, flatness patterns and stochastic texture patterns. In fact, the learned $\lambda_{x,s,\alpha}$ can be either positive for sketch patterns or negative for flatness patterns. (4) It can be used for object detection, alignment, and clustering. (5) It is a probability distribution, so that it can be learned in unsupervised manner. Also, it can be even further adapted to the codebook learning framework proposed in [10].

Limitation and future works: (1) The experiments in this thesis are illustrative and explorative. More empirical and quantitative experiments are needed for better understanding the limitations of the model. (2) Even though the model can capture informative appearance patterns, fitting $p(\mathbf{I})$ in general requires MCMC simulation, which can

be computationally expensive. Further sparsification of $\{\lambda_{x,s,\alpha}, \forall x, s, \alpha\}$ by lasso [19] or matching pursuit [11] can be considered. (3) We assume that the dictionary of the basis functions are given as Gabor wavelets or DoG. It is desirable to learn these basis functions from training images. K-SVD [2] or Olshausen-Field sparse coding [14] can be used to learn Gabor-like bases. (4) The model currently does not incorporate the concept of shape deformations, even though it has taken into account appearance variations. Allowing small perturbations within proper ranges of Gabor and DoG wavelets may be helpful in improving the robustness and the flexibility of the model. This strategy has been successfully used in active basis model [21].

Implementation and Reproducibility

The experiments are run on a PC with Unix operating system, Intel Core i7-3770K CPU@3.50GHz×8, 31.4G memory, and 2 GeForce GTX TITAN GPUs. When fitting a mixture of models, the paralleled computing technique using multiple GPUs is utilized for efficiently learning independent model for each cluster simultaneously. In fitting each single model, the parallel $\tilde{M}$ chains sampling is implemented by tiling a large parameter map $\Lambda = [\lambda, ..., \lambda]_{1\times\tilde{M}}$ with $\lambda = \{\lambda_{x,s,\alpha}, \forall x, s, \alpha\}$, and then sampling directly from $p(\mathbf{I}_{1\times\tilde{M}}; \Lambda)$ by GPU, where the synthesized image $\mathbf{I}_{1\times\tilde{M}}$ would be a tiled image that contains $\tilde{M}$ original-size target images. The tiling order in $\mathbf{I}_{1\times\tilde{M}}$ is consistent with that in Λ. The reason why we sample from a tiled model is as follows: both HMC sampling and updating λ require convolution of images and filters, which is rather time-consuming; while Matlab provides a built-in GPU version of convolution operation that is significantly faster than CPU version. Because GPU consists of thousands of cores to process parallel workloads efficiently, convolving a large image (tiled images) by GPU can bring in more acceleration. All the experimental results reported in this thesis can be reproduced by Matlab and mex-C code that we have posted on the webpage: http://www.stat.ucla.edu/~jxie/iFRAME.html .

REFERENCES

[1] D. H. Ackley, G. E. Hinton, T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, **9**, 147–169. 1985. 2

[2] M. Aharon, M. Elad, and A. M. Bruckstein. The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Trans. Signal Process.*, **5**4, 4311–4322, 2006. 31

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society B*, **3**9, 1–38, 1977. 25

[4] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo, *Physics Letters*, **1**95, 216–222, 1987. 8

[5] A. Gelman and X. L. Meng. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, **1**3, 163–185, 1998. 8

[6] S. Geman, and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Trans. PAMI*, **6**, 721–741, 1984. 8

[7] L.N. Hand, J.D. Finch, *Analytical Mechanics*, Cambridge University Press, 2008. 9

[8] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, **1**4, 1771–1800, 2002. 2

[9] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, **1**8, 1527–1554, 2006. 2

[10] Y. Hong, Z. Si, W. Hu, S. C. Zhu, and Y. N. Wu, Unsupervised learning of compositional sparse code for natural image representation. *Quarterly of Applied Mathematics*, **7**2, 373–406, 2013. 30

[11] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, **4**1, 3397–3415, 1993. 31

[12] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, **1**1, 125–139, 2001. 8

[13] R. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2011. 7, 8

[14] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, **3**81, 607–609, 1996. 31

[15] M. Ranzato and G. E. Hinton. Modeling pixel means and covariances using factorized third-order Boltzmann machines. *CVPR*, 2010. 2

[16] S. Roth and M. Black. Fields of experts. *IJCV*, **8**2, 205–229, 2009. 2

[17] P. Smolensky. Information processing in dynamical systems: foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge, 1986. 2

[18] Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, **4**, 1235–1260, 2003. 2

[19] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B*, **5**8, 267–288, 1996. 31

[20] M. Welling, G. E. Hinton, and S. Osindero. Learning sparse topographic representations with products of student-t distributions. *NIPS*, 2003. 2

[21] Y. N. Wu, Z. Si, H. Gong, and S. C. Zhu. Learning active basis model for object detection and recognition. *IJCV*, **9**0, 198–235, 2010. 31

[22] L. Younes. On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics and Stochastic Reports*, **6**5, 177–228, 1999. 7

[23] S. C. Zhu, Y. N. Wu, and D. B. Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, **9**, 1627–1660, 1998. 1, 2, 4