Generative Modeling of Dynamic Visual Scenes

by

Dahua Lin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

Author .....
Department of Electrical Engineering and Computer Science
August 30, 2012

Certified by .....
John Fisher
Senior Research Scientist
Thesis Supervisor

Accepted by .....
Professor Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

Generative Modeling of Dynamic Visual Scenes

by

Dahua Lin

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Modeling visual scenes is one of the fundamental tasks of computer vision. Whereas tremendous efforts have been devoted to video analysis in past decades, most prior work focuses on specific tasks, leading to dedicated methods to solve them. This PhD thesis instead aims to derive a probabilistic generative model that coherently integrates different aspects, notably appearance, motion, and the interaction between them. Specifically, this model considers each video as a composite of dynamic layers, each associated with a covering domain, an appearance template, and a flow describing its motion. These layers change dynamically following the associated flows, and are combined into video frames according to a Z-order that specifies their relative depth-order.

To describe these layers and their dynamic changes, three major components are incorporated: (1) An appearance model describes the generative process of the pixel values of a video layer. This model, via the combination of a probabilistic patch manifold and a conditional Markov random field, is able to express rich local details while maintaining global coherence. (2) A motion model captures the motion pattern of a layer through a new concept called geometric flow that originates from differential geometric analysis. A geometric flow unifies the trajectory-based representation and the notion of geometric transformation to represent the collective dynamic behaviors persisting over time. (3) A partial Z-order specifies the relative depth order between layers. Here, through the unique correspondence between equivalent classes of partial orders and consistent choice functions, a distribution over the spaces of partial orders is established, and inference can thus be performed thereon.

The development of these models leads to significant challenges in probabilistic modeling and inference that need new techniques to address. We studied two important problems: (1) Both the appearance model and the motion model rely on mixture modeling to capture complex distributions. In a dynamic setting, the components parameters and the number of components in a mixture model can change over time. While the use of Dirichlet processes (DPs) as priors allows indefinite number of components, incorporating temporal dependencies between DPs remains a nontrivial issue, theoretically and practically. Our research on this problem leads to a

new construction of dependent DPs, enabling various forms of dynamic variations for nonparametric mixture models by harnessing the connections between Poisson and Dirichlet processes. (2) The inference of partial Z-order from a video needs a method to sample from the posterior distribution of partial orders. A key challenge here is that the underlying space of partial orders is disconnected, meaning that one may not be able to make local updates without violating the combinatorial constraints for partial orders. We developed a novel sampling method to tackle this problem, which dynamically introduces virtual states as bridges to connect between different parts of the space, implicitly resulting in an ergodic Markov chain over an augmented space.

With this generative model of visual scenes, many vision problems can be readily solved through inference performed on the model. Empirical experiments demonstrate that this framework yields promising results on a series of practical tasks, including video denoising and inpainting, collective motion analysis, and semantic scene understanding.

Thesis Supervisor: John Fisher
Title: Senior Research Scientist

Acknowledgments

Upon the completion of this thesis, it is time to deliver my gratitude to all who have given me the encouragement, support, and love that I need to pursue a PhD at MIT – a journey filled with challenges unprecedented in my life.

First and foremost, I owe my sincere gratitude to my thesis supervisor John Fisher. He led me to a fascinating realm where the beautiful theory of probabilistic modeling and the interesting applications in computer vision meet each other. As my advisor, he gave me total freedom to explore what I was interested in, while providing a lot of valuable suggestions in model formulation, experiment design, and paper revision. I appreciate all his contributions of time and energy to make my academic pursuit productive and rewarding. It was definitely a great pleasure working with him.

I want to express my thanks to Professor Eric Grimson. Eric was the department head of EECS and is now the Chancellor of MIT. In spite of his administrative commitment, he met with me on a regular basis and was always supportive of my research and career development. Instead of offering detailed advices, Eric usually presented questions at a high level, which have been proved to be crucial in keeping me on the right track.

My thanks also go to Professor Alan Willsky. In each semester, Alan held weekly group-lets for both his and John's students. I was fortunate to attend such group-lets and learned a lot from him. His insightful views of various research topics have not only helped me through a number of technical difficulties, but also led me to new perspectives of the problems that I was working on.

I felt honored to be a member of John's research group in CSAIL. The great help from other members of the group has made my professional time at MIT efficient and enjoyable. This group is also the source of discussion and collaboration. Particularly, I would like to thank Jason Chang for his help in resolving technical issues, Donglai Wei for the inspiring discussion and the ECCV paper that we collaborated on, and Randi Cabezas for the preparation of all those wonderful lunches. I am also thankful of other members of both John and Eric's groups: Xiaogang Wang, Xiaoxu Ma,

Chaowei Niu, Gerald Dalley, Biswajit Bose, Zoran Dzunic, Giorgos Papachristoudis, Katie Bouman, and Sue Zheng.

The life at MIT means much more than research. It was my great honor to know many new friends on this campus, who are a constant source of hope and courage. Among these friends, I would like to express my gratitude in particular to Jing Chen, who was always willing to lend her hands to her friends, Jingqing Zhang, a lovely and kind-hearted girl who published the first chemical engineering paper with my name on the author list, Yaodong Zhang, who was able to turn any conversation into a fun, Xiaoqian Jiang, a good friend who helped me a lot in my personal affairs and shared with me a lot of interesting things both inside and outside academia, and Jianxiong Xiao, whose lovely kid brought lots of joy to the lab. I am also indebted to Jingjing Liu, Mengdi Wang, Fei Liang, Ying Liu, Yuan Luo, Yuan Shen, Xi Wang, Yang Cai, Yu Xin, Maokai Lin, and Ce Liu.

Tremendous thanks should be delivered to my family, for my parents who brought me from an infant to a PhD with immense love and support, and for my loving, encouraging, and understanding wife – Leimi, who was always standing by me when I was facing challenges and difficulties, and have sacrificed a lot in support of my academic pursuit. A thousand thanks, my beloved!

Finally, I acknowledge the funding sources that supported my PhD work: (1) Heterogeneous Sensor Networks (HSN), which is supported by the Army Research Office (ARO) Multidisciplinary Research Initiative (MURI) program (Award W911NF-06-1-0076), (2) Stratigraphic Pattern Recognition, which is sponsored by Shell, (3) Nonparametric Representations for Integrated Inference, Control, and Sensing, which is supported by the Defense Advance Research Projects Agency (Award FA8650-11-1-7154), and (4) Nonparametric Bayesian Models to Represent Knowledge and Uncertainty in Decentralized Planning, which is supported by the Office of Naval Research Multidisciplinary Research Initiative (MURI) program (Award N000141110688).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the fundamental goals of computer vision is to derive intelligent systems that can reason about visual scenes, typically captured in the form of images and videos. The past decades have witnessed tremendous efforts towards this goal, resulting in great advancement in a wide range of computer vision topics. However, a substantive amount of prior work is dedicated to specific vision problems, such as object recognition, image segmentation, and motion analysis, leading to substantially different models for describing different aspects of a scene.

This thesis pursues a different approach. Instead of seeking solutions dedicated to particular problems, the primary goal of this thesis is to develop a generative model of dynamic visual scenes that integrates models of different aspects (*e.g.* appearance and motion) into a probabilistic framework. This work is driven by our strong desire to understand the fundamental structures of visual scenes. As Kurt Lewin said,

> *There is nothing more practical than a good theory.*

Whereas problem-oriented approaches can be very successful in accomplishing specific tasks, our view is the advancement of computer vision ultimately relies on deep understanding of the visual scenes, as well as effective means to capture their structures. From a practical standpoint, many applied problems can be readily solved through the inference performed on an integrated generative model. Moreover, as opposed to descriptive and discriminative methods, a generative model also provides greater

flexibility to leverage observations acquired in different ways and take into account various statistical relations.

However, the use of generative models, as opposed to discriminative methods, often comes with additional complications in both model formulation and algorithm design. Therefore, special attention should be paid to making appropriate tradeoffs in order to achieve the desired expressiveness without significantly increasing computational complexity. Through out the entire thesis, we will see, modeling choices made with the careful consideration of such balance.

Generally, visual scenes are complex and far beyond the capacity of a thesis to provide a complete interpretation that takes all relevant aspects into account. This thesis particularly focuses on three key aspects – *appearance*, *motion*, and *composition*, and develops a probabilistic framework that integrates these aspects to give a coherent interpretation of a visual scene. Conceptually, the *appearance* aspect is about what the scene looks like; the *motion* aspect is about how the shapes and positions of the objects in a scene change over time; the *composition* aspect, on the other hand, is about how different parts of a scene are brought together. Generally, these aspects are closely related. For example, motion will cause dynamic changes of appearance, and the compositional structure will greatly influence both the appearance and the perceived motion.

1.1 Questions to be Answered

Towards an integrated model of visual scenes, this thesis tries to address a series of questions as outlined below:

1. *How can we model the appearance?* While humans can perceive objects and regions when looking at an image or a video, what a computer sees is technically no more than a large matrix of pixel values (*i.e.* intensities or colors). The spatial configuration (pattern) of these values constitutes the image's *appearance*. The question here is how to represent these patterns in a way that explains the inherent structure of the visual scene. Generally, the characteristics of patterns

at different scales are substantially different, and should therefore be represented and modeled in different ways.

2. *How can we model the motion occurring in a dynamic scene?* In a dynamic scene, various phenomena can cause changes in appearance. One of the most common causes is *motion*, the change in positions and shapes of objects. There has been extensive study on motion analysis in past decades, much of which aims at accurate estimation of local movement of individual objects or points. However, our sense of motion in many natural scenes is reflected by the changes over a region or by the collective behavior of a group of objects, which we refer to as the *collective dynamics*. Effective analysis of the collective dynamics (e.g. revealing the underlying coherence between the motion of different objects) is often the key to the understanding of visual scenes.

3. *How can we handle concurrent entities in a visual scene?* It is not uncommon in natural scenes that multiple entities (e.g. objects and flows) are observed at the same time. Each entity may have its own appearance and behavior. When a natural scene is projected onto an image plane, occlusion may occur. Objects occluded by others are only partially observed, further complicating analysis. It is possible to rely on a three-dimensional model to resolve this issue, but doing so generally requires 3D reconstruction, which in itself is a nontrivial problem. As we will see, a layered representation, which captures the relative depth order between objects instead of their 3D positions, is often sufficient to resolve most of the ambiguities resulting from occlusions, and can be inferred more reliably and efficiently.

4. *How can we model the relations between appearance and motion?* As mentioned, a dynamic visual scene is characterized by both the spatial structure and the temporal dynamics, which are respectively captured by the appearance and the motion. These two aspects are not completely independent. Instead, they are closely coupled with each other, and what we see in a video is actually the compound effect of both. While separate study of each aspect is useful, it is

also very important to understand how they relate to each other. Such spatial-temporal relations can be exploited to improve the analysis of videos and help other video-related tasks.

5. *How can we handle model complexity?* Tradeoffs between expressiveness and complexity has been one of the central themes of machine learning and related fields such as computer vision. Mixture models, which are often used to capture complex distributions, are employed in many vision models. An important issue here is how to determine the number of components in a mixture model (*i.e.* the model order). In a dynamic setting (*e.g.* video analysis), the phenomena of interest may evolve over time. Modeling such phenomena generally requires a model which is able to change its order adaptively. Formulating and estimating models with dynamic complexity is a challenging problem.

1.2 The Overall Scene Modeling Framework

The first step of visual scene modeling is to choose a specific way to construct the model. In general, dynamic scenes can be very complex. To effectively model such scenes, we have to make simplified assumptions, emphasizing key aspects, while deliberately neglecting the others. First of all, we have to decide the basic structure of the model. Here, several questions arise:

- *What are the basic components?*
- *How do the components interact?*
- *How do they evolve over time?*

Generally, there are three approaches to scene modeling, with different answers to these questions, which we will briefly review below.

1.2.1 Three Approaches to Scene Composition

Existing approaches to scene modeling can be roughly classified into three categories, according to the ways they model the compositional structure of a scene.

1. **Segmentation-based Models.** Segmentation is widely used in analyzing images comprised of multiple regions. Models in this category describe an image as composed of multiple disjoint regions called *segments*. The appearance within each segment has relatively consistent characteristics, while such characteristics in neighboring regions may be remarkably different.

   A segmentation-based model typically comprises a set of appearance models, each for a particular region, and a model that incorporates prior knowledge about the segmentation itself (*e.g.* spatial continuity and the smoothness of the segment boundaries). Such approaches aim to capture common visual characteristics within each region while allowing substantial variation across the boundaries.

   Despite its utility in image analysis, several fundamental problems limit the use of segmentation-based approaches in dynamic contexts, especially when occlusion occurs. First, an object can be divided into disconnected segments in different ways, sometimes complicating the correspondence between segments across video frames. Second, segments moving towards each other and then overlapping would lead to a "conflict of explanation", while segments moving away may leave part of the image covered by no region. These problems stem from the occlusion occurring when a three dimensional scene is projected onto a two dimensional view.

2. **Three Dimensional Models.** A three-dimensional (3D) model describes a visual scene within a 3D coordinate system, and observed images of the scene as projections onto 2D image planes. By maintaining the 3D positions of objects, the ambiguities encountered by segmentation-based models can be effectively addressed.

Generally, the process of obtaining a 3D visual model from observed images is called *3D reconstruction*, which in itself is nontrivial. It has long been an active topic in computer vision. Typically, 3D reconstruction requires stereopsis (*a.k.a.* binocular vision), or relies on knowledge about the geometric relations between the scene and the camera to recover the scene structure. However, in practice, many videos of interest are captured impromptu, or without using a calibrated camera, making it difficult to obtain a 3D model reliably. In addition, the computational complexity required to estimate and maintain a 3D model is often higher than that for methods based on 2D image models.

3. **Layered Models.** The aforementioned difficulties motivate researchers to explore more effective approaches to generic video modeling. *Layered video models*, introduced in Wang and Adelson's pioneering work [109], have become increasingly popular for dynamic scene modeling.

   In general, a layered model describes an observed image of a scene as a superposition of multiple layers, each corresponding to an object or a set of objects with coherent behavior. One major difference that distinguishes a layered model from a segmentation-based model is that a layered model allows different layers to overlap and explicitly takes into account the occlusion relations between them.

   Instead of trying to estimate the depth map as in methods using 3D models, a layered model relies on occlusion reasoning, which is generally much easier, especially when the scene is captured with a single camera.

Based on the considerations above, we chose to construct the model of dynamic scenes using a layered structure. Next, we will outline the overall formulation of this model, and identify the key components.

1.2.2 A Layered Model of Dynamic Scenes

A layered model considers a video as a composite of multiple dynamic layers. It is difficult to characterize a *layer* in general, as its meaning often depends on specific

Figure 1-1: This figure shows several cases where the simplified assumptions underlying the layered model are violated. (a) shows a girl behind a boy, while her hands are in front of him. If we model them as two layers, then there is no definite depth order between them. (b) shows two kids playing basketball. While they can be modeled as two dynamic layers, their behavior are not independent. (c) show a soldier in a battlefield. His behavior may be influenced by many factors, and simply modeling it as a Markov chain may overly simplify the real world complication.

context (*e.g.* the type of the scene, and the scale at which the analysis is being performed). Generally, a layer may correspond to an object, a group of objects with consistent behaviors that are spatially close to each other, or a part of the scene with coherent dynamics.

The study presented in this thesis is based on the layered video model as described below. Consider a video with n layers, denoted by $L_1, \ldots, L_n$. Each layer (say the i-th one L_i) is associated with (1) a *covering domain* D_i that specifies sub-region in the image plane *physically* covered by the layer, (2) an *appearance template* A_i that describes the visual content of the layer (or technically, the spatial pattern of the pixel values in the layer), and (3) a *flow* F_i that describes the motion.

For a dynamic scene, the covering domain, the appearance template, as well as the flow associated with a layer may each vary over time. To reflect this, we use $D_i^{(t)}, A_i^{(t)}$, and $F_i^{(t)}$ to respectively denote the versions of these components at time t. The changes occurring in different aspects are related. For example, the changes in appearance or covering domain may be influenced by motion. To capture such relations, we formulate the dynamics of these aspects jointly through the conditional

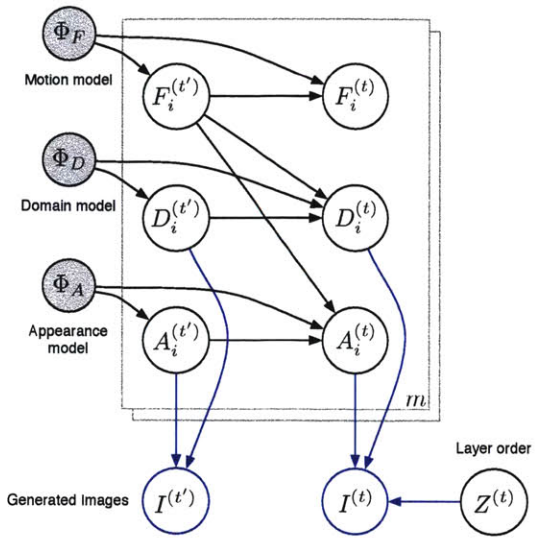


Figure 1-2: The graphical representation of the layered visual model. As shown in this figure, the framework is comprised of m layers, each associated with a motion model, a domain model, and an appearance model. These aspects are assumed to be independent a priori, and dynamic evolution of each aspect is assumed to follow a Markov chain. Given the characterizations of these aspects, a video frame $I^{(t)}$ can be generated according to the relative depth order between layers, which is denoted by $Z^{(t)}$. The arrows in blue color highlight the factor that formalizes the generation of video frames conditioned on latent states of these aspects.

distribution as below:

$$p(D_i^{(t)}, A_i^{(t)}, F_i^{(t)} | D_i^{(t-1)}, A_i^{(t-1)}, F_i^{(t-1)}; \Phi_D, \Phi_A, \Phi_F) = \\ p(D_i^{(t)} | D_i^{(t-1)}, F_i^{(t-1)}, \Phi_D) \cdot p(A_i^{(t)} | A_i^{(t-1)}, F_i^{(t-1)}, \Phi_A) \cdot p(F_i^{(t)} | F_i^{(t-1)}, \Phi_F). \quad (1.1)$$

This formula comprises three factors:

1. The factor $p(D_i^{(t)} | D_i^{(t-1)}, F_i^{(t-1)}, \Phi_D)$ describes how the covering domain of the i-th layer evolves over time. Φ_D is the parameter for this factor. In general, the evolution of the domain D_i may depend on F_i, the motion of the i-th layer.

2. The factor $p(A_i^{(t)} | A_i^{(t-1)}, F_i^{(t-1)}, \Phi_A)$ describes changes in appearance, controlled by the parameter Φ_A. Such changes may also depend on the underlying motion.

3. The factor $p(F_i^{(t)} | F_i^{(t-1)}, \Phi_F)$ describes how the motion field itself changes over time. Here, it is assumed that the dynamics of the motion field is independent

of the appearance a priori.

To establish a complete probabilistic formulation, we also need a prior distribution over these components for the initial frame, as

$$p(D_i^{(0)}, A_i^{(0)}, F_i^{(0)} | \Phi_D^0, \Phi_A^0, \Phi_F^0) = p(D_i^{(0)} | \Phi_D^0) p(A_i^{(0)} | \Phi_A^0) p(F_i^{(0)} | \Phi_F^0). \tag{1.2}$$

As the motion only affects what we may observe in the next frame, the initial prior of the covering domain and the appearance does not depend on the motion.

Given all the layers, each observed video frame can be generated through superposition. Note that the covering domains of different layers can overlap, some regions may be covered by more than one layer. In such a region, only the layer at the top is visible, while others are occluded. This model uses a partial order to capture the relative depth order between layers such that the top layer of each region can be readily determined. Let Z denote this partial order and x be a pixel location, then the visible pixel value at x is given by

$$I^{(t)}(x) = L_{\iota(x)}^{(t)}(x), \quad \iota(x) = \max_Z \{i : x \in D_i^{(t)}\}. \tag{1.3}$$

Here, $\iota(x)$ denotes the index of the layer that is visible at x, which is set to be the maximum among all the layers that cover x. Note that the maximum here is with respect to the depth order Z, implying that it corresponds to the top layer. In addition, we use $L_i^{(t)}(x)$ to denote the pixel value at x of the layer L_i at time t. Hence, following a (partial) depth order, all layers can be combined in a consistent way into video frames.

We obtain a joint model, as shown in Figure 1-2, by integrating the priors for individual layers with the factor above that describes how layers combined to produce observe images. Note that a probabilistic model expressed in form of a graph is often referred to as a *graphical model*, for which Chapter 2 provides a detailed treatment.

1.2.3 Discussion on Modeling Choices

Several simplifying assumptions underlie the model established above.

1. A visual scene can be decomposed into a superposition of several layers, and that these layers follow a consistent depth order. This model enforces a strong constraint that precludes some cases, such as those illustrated in Figure 1-1(a), where part of a layer is below another, while the other part is above. In addition, the layers are considered to be non-transparent, meaning that the occluded part of a layer will be completely invisible. It is possible to further extend the model further to explain such scenes, but this is outside the scope of this thesis.

2. The appearance and dynamics of different layers are assumed to be independent. This might seem to be an overly simplified assumption, as behaviors of objects coexisting in the same scene might interact with each other in various ways (an example is shown in Figure 1-1(b)). While understanding such interactions might be in the interest of some high-level applications, such as behavior analysis, it is reasonable to ignore them in constructing a lower-level vision systems, because the primary goal here is to derive an intermediate representation of the appearance and motion. For a problem where such interactions play a crucial role, one can build an interaction model on top of the vision model being discussed in this thesis.

3. The dynamics of each layer is modeled as a first-order Markov chain. This assumption ignores some real world complexities where the behavior of an object may depend on many other factors in addition to how it behaves at the previous time step, as illustrated in Figure 1-1(c). Again, to keep the model simple, we chose to focus on the aspects directly pertinent to vision problems. It is possible to develop higher level reasoning methodologies on top of the framework developed by this thesis.

1.2.4 Main Aspects

In the visual modeling formulation outlined above, we can identify several key aspects and how they relate to each other. These aspects are summarized as follows.

1. **Appearance.** The appearance model specifies the spatial structure of the visual scene. In particular, the pixel values in each layer L_i are captured by an appearance template A_i. The associated appearance model (with parameter Φ_A) provides a generative prior $p(A_i|\Phi_A)$ over such appearance templates, specifying how the pixels are distributed, how their values relate to each other, and what the spatial structures are.

2. **Motion.** The dynamics of a visual scene is mainly captured by the motion model. At the heart of the motion model is an intermediate representation of the motion, called *flows*. Particularly, each layer L_i is associated with a flow, denoted by F_i. In addition, the motion model also specifies the prior distribution of the flows $p(F_i^{(0)}|\Phi_F^0)$ and how flows evolve over time $p(F_i^{(t)}|\Phi_F)$.

3. **Layer Order.** The composition of layers into video frames depends on the relative depth order between layers. A layer can be occluded by the one above it when they overlap. This relation between layers can be formalized as a partial order, over which we can define a prior distribution for Bayesian inference.

Whereas we do not develop a complete interpretation of visual scenes in this thesis, we view the framework and methodology presented in this thesis as part of an overall effort towards that ultimate goal. Putting this thesis in a broader perspective of computer vision research, we acknowledge that some important problems are not covered by the three aspects above (*e.g.* the modeling of layer domains, the relations between layers from different views, and semantic implications of the characterizations for appearance and motion). Still, this thesis contributes to the advancement of computer vision in several important ways:

- It demonstrates the utility of generative models in computer vision, and illustrates how they can be formulated to solve vision problems through the devel-

opment of specific models to describe appearance, motion, and layer order.

- The models studied in this thesis can be extended, modified, or adapted to solve other problems that we do not explicitly consider. For example, the appearance model can be generalized to describe color images or textures on 3D surfaces, and the motion model can be adapted by incorporating additional structure when applied to a specific context. Furthermore, these models can be used in combination with other vision models to derive more sophisticated frameworks or to address more complex issues.

- A series of methods, such as dynamic nonparametric models and methods to sample from combinatorial spaces, has been developed to address specific challenges in vision problems. The use of these methods, however, is by no means restricted to the applications discussed in this thesis. They can be applied, sometimes with modification, to solve other problems, including many outside the realm of computer vision.

1.3 The Organization of the Thesis

Thus far, a high-level structure of the scene model has been specified. However, many interesting but challenging questions are yet to be answered: *e.g.* how to represent appearance and motion, how to define the prior distribution and conditional distribution, how to estimate the model parameters, and how to perform inference over this model. Most of the work presented in this thesis tries to answer these questions, as we shall see in the following chapters.

The remaining part of this thesis is organized as follows. Chapter 2 briefly reviews the advancement of visual modeling in past decades with an aim to provide a retrospective view of of ways in which recent progress of this field is influenced by the prevalent use of probabilistic models. This chapter also covers the basic concepts for probabilistic modeling, such as Bayesian networks and Markov random fields, as well as the basic tools for learning and inference, including mean field approximation,

belief propagation, and Monte Carlo sampling. The materials in this chapter lay the theoretical foundation for later discussions.

Starting from Chapter 3, we present the vision models in detail. Particularly, this chapter describes a new image prior for appearance modeling. The development of the new image model is motivated by the key insight that the effectiveness of an image model, to a large extent, hinges on its capability of capturing local pixel patterns and maintaining the coherence of local structures. To improve on these aspects, we develop a new generative model of images, which integrates a patch manifold to capture local patterns and a conditional Markov random field to enforce coherence across patches. We also derive algorithms to estimate the model parameters from a set of training images. It is important to note that with this model, a set of low-level vision problems, including image denoising and inpainting, can be readily solved via the inference performed based on a joint model that combines this prior with a specific measurement process.

The results obtained by applying this model to image recovery are also be presented and analyzed.

Chapter 4 discusses motion models. As an important area of computer vision, motion estimation has been extensively studied. Prior work on motion modeling and estimation has focused on tracking and optical flow. However, in many scenes, the overall sense of dynamics is reflected by the collective movement of large groups of objects/people, or by the motion over a region, which we refer to as the *collective dynamics*. The primary goal here is to develop an effective framework to characterize and estimate such collective dynamics. In this chapter, we introduce the notion of *geometric flow*, a concept originating from differential geometry, which is able to capture motion patterns that persist (or smoothly evolve) over time. Subsequently, we derive a vector space of flows by exploiting the intrinsic connection between Lie groups and Lie algebras. A stochastic flow model is then developed on top of this, with which flow parameters can be efficiently inferred from different types of observations, including tracks of key points and continuous changes in image appearance. Application of the new motion model in several different contexts is also presented in

this chapter.

In both appearance and motion modeling, mixture models, which bring together a set of component models to approximate complex distributions, play a crucial role. Traditional study of mixture models focuses mainly on the task of fitting a mixture to a given set of data. However, for video analysis, mixture models may need to evolve over time so as to adapt to the changes of the observed scene. This motivates the development of tools to construct dynamic mixture models in Chapter 5. Specifically, this chapter first reviews *finite mixture models*, which are widely used in practice, and *Dirichlet process mixture models (DPMM)* – a new way to construct mixture models that allow an indefinite number of mixture components. The key challenge of using DPMMs in a dynamic context is to allow the mixtures to evolve while maintaining strong dependencies between them. To solve this problem, we develop a new approach to constructing *dependent Dirichlet processes*, which, unlike classic methods such as the Chinese Restaurant Processes and Stick Breaking Processes, explicitly exploits the intrinsic connection between Dirichlet and Poisson process and the concept of complete randomness. Upon this construction, several primitive operations are derived, allowing the mixture model to change in various ways. In addition to theoretical analysis, this chapter also demonstrates the utility of this new construction in flow modeling and video interpretation, as well as applications outside the vision domain.

Chapter 6 considers the layered structure of the model. As mentioned, layers can overlap and thus a partial order is needed to keep track of the relative depth order between them. This chapter studies various properties of a partial order, such as sufficiency, minimality, and identifiability, and how they relate to visual modeling. Based on this analysis, we establish an efficient representation using directed graphs. In this chapter, we also discuss the methods for inferring partial depth order from observed scenes. Generally, MAP estimation of the optimal partial order is NP-hard, and the combinatorial constraints on partial orders also lead to great difficulties in devising sampling schemes (*e.g.* the underlying space can be disconnected due to the constraints). To address this problem, we develop a novel method to efficiently sample

from a constrained combinatorial space by constructing an augmented Markov chain with improved mixing performance through the introduction of bridging nodes.

Finally, Chapter 7 concludes the entire thesis, summarizing the key aspects of the dynamic scene model and the probabilistic modeling techniques developed to address some of the challenges arising therefrom. In this chapter, we discuss several directions that merit further study.

Chapter 2

Theoretical Background

The visual scene model developed in this thesis is a Bayesian probabilistic model, which integrates the models of different aspects, notably the appearance model, the motion model, and the model of layer orders, into a joint formulation.

The development of these models heavily relies on various tools of probabilistic modeling and inference, which, in particular, includes graphical models, exponential family distributions, variational inference, and Monte Carlo sampling.

In this chapter, we first review the basic concepts graphical models, a graphical representation of joint distributions that indicates dependencies between variables through edges (or arrows). With a probabilistic graphical model, one can estimate the most probable value of the variables of interest (or the posterior distributions over them) through inference, conditioned on the variables whose values are known.

However, performing exact inference over a graphical model can be computationally intractable in practice. In such case, one can resort to techniques that can perform the inference approximately with reasonable complexity. This chapter also gives a brief exposition of these approximate inference techniques, including variational inference and Monte Carlo sampling.

Note that the contents covered by this chapter are not the contribution of this thesis. They are mostly from existing work. Nonetheless, this chapter is indispensable, as it lays the theoretical foundation for the development in other chapters.

2.1 Probabilistic Graphical Models

Graphical models, through the combination of graph theory and probability theory, provides a powerful and elegant means to formulate probabilistic models. The key idea underlying graphical models is *factorization*: a joint distribution represented by a graphical model can generally be factorized according to the structure of the underlying graph. Through such factorization, model estimation and inference can often be greatly simplified.

2.1.1 Basic Concepts of Graphical Models

A graphical model defines a family of probabilistic distributions based on graph, of which each vertex is associated with a random variable and edges are used to indicate the statistical dependencies between variables.

First of all, we set up the notations. Consider a graph $G = (V, E)$. The random variable associated with a vertex $v \in V$ is denoted by X_v, which can take values in some space $\mathcal{X}_v$. A lower-case letter (say $x_v \in \mathcal{X}_v$) denotes a particular value assigned to X_v. For a subset of vertices A, X_A denotes the collection of random variables as $X_A := (x_v, v \in A)$, and x_A denotes a particular assignment to X_A.

Bayesian Networks

The graphical models associated with an *acyclic directed graph (DAG)* is often called a ***Bayesian network***. For such a graph, if there is an edge $s \to t$, then s is called a *parent* of t, while t is called a *child* of s. Generally, a vertex may have multiple parents and multiple children. For a vertex v, the set of its parents is denoted by $\pi(v)$, and the set of its children $ch(v)$. A graphical model where each node (except for the *root node*) has exactly one parent is called a ***tree model***.

A Bayesian network defines a family of conditional distributions for each vertex v, as $p_v(X_v|x_{\pi(v)})$, which describes the distribution of X_v conditioned on the values assigned to its parents. When v is the root, $\pi(v) = \emptyset$, this conditional distribution reduces to a prior distribution as $p_v(X_v)$.

Along the graph, the joint likelihood of all variables associated with the graph can be factorized into the product of these conditional factors, as

$$p(x_V) = \prod_{v \in V} p(x_v | x_{\pi(v)}). \tag{2.1}$$

It can be easily verified that with this joint formulation, the conditional likelihood $p(x_v|x_{\pi(v)})$ is exactly equal to the value given by $p(X_v|x_{\pi(v)})$.

Markov Random Fields

The graphical models associated with an *undirected graph* is often called a ***Markov random field (MRF)***, which factorizes according to *cliques*. Here, a ***clique*** C is defined to be a fully connected subset of vertices.

Generally, an MRF considers a set of cliques $\mathcal{C}$ ($\mathcal{C}$ may be a proper subset of all cliques) and defines a *(potential function* for each clique $C \in \mathcal{C}$, as $\psi_C : \bigotimes_{v \in C} \mathcal{X}_v \to \mathbb{R}^+$, which maps each particular value assignment of clique C to a nonnegative real number, which reflects *how compatible the assignment is with the model.*

With the clique potentials, the joint likelihood of all variables can be written as

$$p(x_V) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C), \tag{2.2}$$

where Z is a normalization constant, given by

$$Z = \int_{x_V} \prod_{C \in \mathcal{C}} \psi_C(x_C) \mu_0(dx_V). \tag{2.3}$$

Here, μ_0 denotes the base measure of the joint space. For continuous variables, it is the Lebesgue measure; while for discrete variables, it is the counting measure.

For a general MRF, the potential functions ψ_C need not be pertinent to any marginal or conditional distributions over the cliques. The only restrictions to ψ_C is that they should be non-negative and their product is integrable (*i.e.* Z is finite).

The maximum cardinality of a clique is called the ***order*** of the MRF. It is easy

to see that in a first-order MRF, all variables are independent, which is not very interesting. Second-order MRFs have been widely used in practice, due to its simplicity. A second-order MRF consists of two types of potentials, the ones over single variables and those over pairs of connected variables. Generally, the formulation of a second-order MRF can be written in the following form:

$$p(x_V) = \frac{1}{Z} \prod_{v \in V} \psi_v(x_v) \prod_{\{u,v\} \in E} \varphi(x_u, x_v). \tag{2.4}$$

From Bayesian networks to Markov random fields

It is useful to note that a Bayesian network can be considered as a special case of a Markov random field.

Specifically, a Bayesian network as formulated in Eq.(2.1) can be treated as a Markov random field defined on a set of cliques as

$$\mathcal{C} = \{C_v := \{v\} \cup \pi(v) | v \in V\}.$$

Here, the potential function associated with the clique C_v is simply the conditional pdf of X_v, as

$$\psi_{C_v}(x_v, x_{\pi(v)}) \triangleq p(x_v | x_{\pi(v)}),$$

and the normalization constant Z equals 1. However, in general, MRF cannot be converted to a Bayesian network (except for some special cases).

From a graph theoretical perspective, this re-formulation turns a directed graph into an undirected graph, by (1) converting all directed edges to undirected edges, and (2) adding undirected edges between pairs of parents of each vertex (if they are not connected). This process is called ***moralization***.

Factor graph

Both Bayesian networks and Markov random fields can be represented uniformly as *factor graphs*. Different from a graphical model introduced above, a factor graph is a

bipartite graph consisting of two types of nodes: *variable nodes*, each associated with a random variable, and *factor nodes*, each associated with a factor (*e.g.* conditional pdfs and clique potentials). Each edge in this graph connects between a factor and a variable involved in that factor.

With a factor graph, the joint distribution over all variables can be written in form of a product of factors, as

$$p(x_V) = \prod_{j=1}^{m} f_j(x_{S_j}).$$

Here, V denotes the set of all variable nodes, j is used as the index of factors, and S_j denotes the subset of variables involved in the j-th factor.

As we shall see later, the notion of factor graph will greatly simplify the description of *belief propagation*, a general message passing algorithm for computing marginals over a graphical model.

2.1.2 Conditional Independence

Probabilistic graphical models can also be characterized through conditional independence between random variables, also known as the *Markov properties*.

Here, we briefly review the notion of *conditional independence*.

Definition 2.1 (Conditionally independent variables). *Let X, Y, Z be random variables, with a joint distribution $p(X, Y, Z)$. If for any assignment x, y, z,*

$$p(x, y|z) = p(x|z)p(y|z),$$

we call X and Y are ***conditionally independent*** *given Z (or X and Y are independent conditioned on Z), denoted as $X \perp Y|Z$.*

For a general graphical model defined on a graph G, the conditional independence is determined by the structure of G. However, the way to determine conditional independence for Bayesian networks is different from that for Markov random fields.

Conditional independence between variables of a Bayesian network can be determined by examining the *d-separation.*

Definition 2.2 (D-separation). *Let G be a directed graph and A, B, C be disjoint subsets of vertices, any trail p satisfying either of the following conditions is said to be* ***d-separated*** *by C:*

1. *p contains a chain $u \to m \to v$ with $m \in C$;*
2. *p contains a chain $u \leftarrow m \leftarrow v$ with $m \in C$;*
3. *p contains a fork $u \leftarrow m \to v$ with $m \in C$;*
4. *p contains an inverted fork $u \to m \leftarrow v$, where m is neither in C nor a descendant of any vertex in C.*

A and B are said to be ***d-separated*** *by C is any trails between a vertex in A and a vertex in B is d-separated.*

Proposition 2.1. *Given a Bayesian network defined on an acyclic directed graph G. Let A, B, C be disjoint subsets of vertices. Then $X_A \perp X_B | X_C$ if A and B are d-separated by C.*

Conditional independence between variables of an MRF can be determined much more easily, by examining normal graph separation.

Proposition 2.2. *Given a Markov random field defined on an undirected graph G. Let A, B, C be disjoint subsets of vertices. Then $X_A \perp X_B | X_C$ if any paths between a vertex in A and a vertex in B passes through some vertex in C.*

Enumerating all possible choices of subsets A, B, and C results in a list of assertions of conditional independence. It can be shown that the collection of conditional independence obtained by d-separation analysis on a Bayesian network is equal to that obtained through the analysis on the MRF derived by moralization.

Equivalence of characterization

We have discussed two characterizations of probabilistic graphical models: *factorization* and *conditional independence*. The *Hammersley-Clifford theorem* below, which is a fundamental result of graphical model theory, establishes the equivalence of these two characterizations.

Theorem 2.1 (Hammersley-Clifford theorem). *Let $G = (V, E)$ be an undirected graph, of which each vertex v is associated with a random variable X_v taking value in $\mathcal{X}_v$. Suppose $p(x_V) > 0$ for every $x_V \in \bigotimes_{v \in V} \mathcal{X}_v$* ***(positivity condition)****, then the following statements are equivalent:*

1. *p satisfies* ***local Markov property****: for each vertex $v \in V$, X_v is independent of all other variables conditioned on its neighbors, as*

$$X_v \perp X_{V \setminus (\{v\} \cup \mathcal{N}(v))} | X_{\mathcal{N}(v)}, \tag{2.5}$$

   where $\mathcal{N}(v)$ denotes the set of neighbors of v (excluding v itself).

2. *p satisfies* ***global Markov property****: let A, B, and C be disjoint subsets of vertices, such that A and B are separated by C (i.e. every path between A and B passes through some vertex in C, or there is no path between A and B), then $X_A \perp X_B | X_C$.*

3. *p can be factorized according to the cliques of the graph. Particularly, let $\mathcal{C}$ be the set of all maximal cliques of G, then p can be written as*

$$p(x_V) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C). \tag{2.6}$$

A similar result holds for Bayesian networks, where the only modification is using d-separation instead of graph separation in describing the Markov properties.

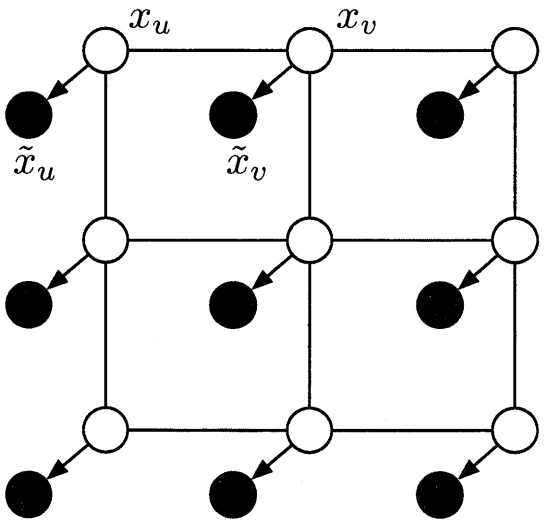


Figure 2-1: This illustrates an MRF model for image modeling, where each node, as denoted by x_u and x_v, corresponds to a pixel of the image. Edges are incorporated to link between neighboring nodes to enforce smoothness among them. In addition, to account for the measurement noise, it associates each pixel an additional node, as denoted by $\tilde{x}_u$ and $\tilde{x}_v$, to represent the actual observation, which are assumed to be generated from the underlying pixel by adding noise.

2.1.3 Example Applications in Computer Vision

Probabilistic graphical models have been widely used in computer vision in the past decade. Here, we use two simple examples to illustrate its use in visual modeling: *MRF for image modeling* and *HMM for dynamic modeling*. These examples can be considered as simplified versions of the models developed in later chapters.

MRF for image modeling

Modeling images has been a central topic of computer vision. A classic approach to image modeling is to use a Markov random field to capture the local relations between neighboring pixel values.

Here, we described a classic MRF formulation for image modeling. As shown in Figure 2-1, this model is comprised of a grid of nodes, where each node X_v is a random variable representing the value of a pixel. In natural images, neighboring nodes tend to have similar values. To reflect this intuition, edges are introduced

between neighboring nodes, giving rise to a set of pair-cliques, each associated with a compatible potential function φ_{uv}. A popular choice of the compatible potential is given by

$$\varphi_{uv}(x_u, x_v) = \exp\left(-\frac{\theta_{uv}}{2}(x_u - x_v)^2\right). \tag{2.7}$$

Here, the weight θ_{uv} reflects how much we think the pixel values x_u and x_v should be close to each other a priori. Combining all these potentials results in an MRF as

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z}\prod_{\{u,v\}\in E}\varphi_{uv}(x_u, x_v) = \frac{1}{Z}\exp\left(-\frac{1}{2}\sum_{\{u,v\}\in E} w_{uv}(x_u - x_v)^2\right). \tag{2.8}$$

Here, E is the set of undirected edges between neighboring nodes.

In practice, the measurement process often introduced noise, and consequently, the observed value is a noisy perturbation of the actual pixel value x_u, which we denote by $\tilde{x}_u$. Taking this into consideration, we can get a joint formulation that generates both the actual pixels and the observed pixel values, as follows

$$p(\mathbf{x}, \tilde{\mathbf{x}}|\boldsymbol{\theta}) = p(\mathbf{x}|\boldsymbol{\theta})\prod_{v\in V} p(\tilde{x}_v|x_v). \tag{2.9}$$

In vision literatures, it is a common practice to assume the noise added to the pixels is white noise, as

$$\tilde{x}_v \sim \mathcal{N}(x_v, \sigma^2).$$

Here, σ^2 is the noise variance.

HMM for dynamic modeling

In modeling dynamic scenes, the temporal relations that describes how things evolve over time plays a central role. In general, such relations can be captured using a *Hidden Markov model.*

Specifically, consider a sequence of video frames, respectively described by feature vectors $x_0, x_1, \ldots, x_T$. At each time step, the generation of the feature vector x_t is controlled by an internal state vector z_t. The content of z_t depends on spe-

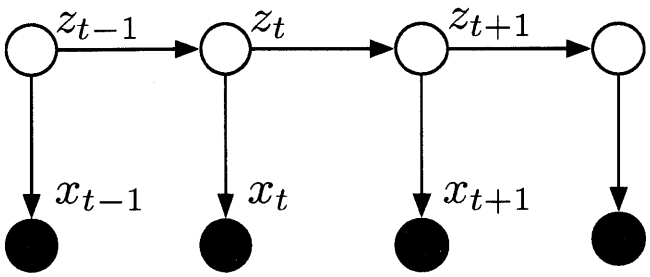


Figure 2-2: This illustrates an HMM model for modeling dynamic scene. The observed scene x_t is associated with an internal state z_t that can evolve over time. The directed edges from each state to the next capture the temporal relations between consecutive states. In addition, under this model, the observed scenes are completely determined by the internal states, which are independent from each other, when the internal states are given.

cific problems, which, for example, may contain shape parameters, kinematic status, and object locations, etc. A ***Hidden Markov Model*** is established based on two assumptions:

1. The internal states constitute a Markov chain as

$$p(z_0, \ldots, z_T) = p(z_0) \prod_{t=1}^{T} p(z_t|z_{t-1}). \tag{2.10}$$

2. The observed features are conditionally independent of each other given the internal states, as

$$p(x_0, \ldots, x_T|z_0, \ldots, z_T) = \prod_{t=0}^{T} p(x_t|z_t). \tag{2.11}$$

Together, the joint formulation that generates both internal states and the observed features can be written as

$$p(x_0, \ldots, x_T; z_0, \ldots, z_T) = p(z_0) \prod_{t=1}^{T} p(z_t|z_{t-1}) \prod_{t=0}^{T} p(x_t|z_t). \tag{2.12}$$

The graphical representation is shown in Figure 2-2, which is a Bayesian network with tree-structure.

2.1.4 Exact Inference

Given a probability distribution p defined as a graphical model, one can solve the following problems:

1. Evaluate the likelihood of observed data;
2. Compute the conditional distribution $p(X_U|x_O)$. Here, O denotes the set of vertices whose values x_O are known (or observed), and U denotes the set of vertices of interest.
3. Compute the most probable value of X_U conditioned on x_O, *i.e.* solve $\hat{x}_U = \text{argmax}_{x_U} p(x_U|x_O)$.

The process of solving one or more of these problems is often referred to as ***probabilistic inference*** (or simply ***inference***).

Performing exact inference upon a generic graphical model is challenging, and in most cases, computationally intractable. However, for tree models, including MRFs on graphs without loops or Bayesian networks of which each vertex has at most one parent, the inference of marginal distribution of each variable can be solved efficiently through an algorithm with recursive message passing, which is also known as the *sum-product algorithm.*

Sum-product algorithm

Consider an MRF defined on a tree $T = (V, E)$. Clearly, it is a second-order model. Designating any one vertex $s \in V$ as the root results in a rooted tree, where each of other vertices (say v) have one parent, denoted by $pa(v)$. Then the joint distribution can be written as

$$p(x_V) = \frac{1}{Z} \prod_{v \in V} \psi_v(x_v) \prod_{v \in V \setminus \{s\}} \varphi_v(x_{pa(v)}, x_v). \tag{2.13}$$

To simplify the following discussion, we focus on discrete variables here. For each

vertex u, we define a function $Q_u : \mathcal{X}_u \to \mathbb{R}^+$ as

$$Q_u(x_u) = \psi_u(x_u) \sum_{x_{D(u)}} \prod_{v \in D(u)} \psi_v(x_v) \prod_{v \in D(u)} \varphi(x_{pa(v)}, x_v). \tag{2.14}$$

Here, $D(u)$ denotes the set of all descendants of u (excluding u itself). Then, the marginal distribution of the root variable X_s is given by

$$p_s(x_s) = \frac{1}{Z} Q_s(x_s), \quad \text{with } Z = \sum_{x_s \in \mathcal{X}_s} Q_s(x_s). \tag{2.15}$$

Evaluation of Q_u directly following Eq.(2.14) is generally intractable. However, this can be accomplished much more efficiently in a recursive fashion.

First, for leaf vertices (*i.e.* those without chidren), Q_u reduces to

$$Q_u(x_u) = \psi_u(x_u). \tag{2.16}$$

For other vertices, Q_u can be written in terms of the function value of u's children, as

$$Q_u(x_u) = \psi(x_u) \prod_{v \in ch(u)} \sum_{x_v \in \mathcal{X}_v} \varphi_v(x_u, x_v) Q_v(x_v). \tag{2.17}$$

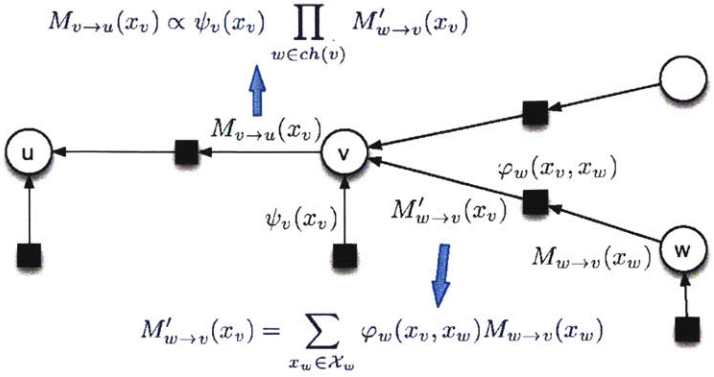
With this recursive formula, the evaluation of all Q_u functions can be done, starting from leaf vertices, recursively upward until Q_s is evaluated.

When $p_s(x_s)$ is derived, the marginal distribution of other nodes can be easily computed through a downward sweep, following the formula below

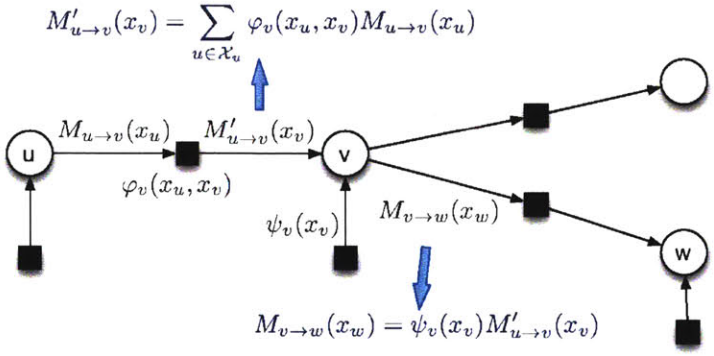
$$p_v(x_v) \propto \psi_v(x_v) \sum_{x_u \in \mathcal{X}_{pa(v)}} p_{pa(v)}(x_u) \varphi(x_u, x_v). \tag{2.18}$$

Belief propagation

The sum product algorithm presented above can be reformulated as a message passing process along a factor graph, as demonstrated in Figure 2-3. Here, messages are exchanged between factors and variables. In particular, Q_v can be considered as the



(a) Upward message passing from leafs to root



(b) Downward message passing from root to leafs

Figure 2-3: This illustrates the procedure that recursively evaluates the marginals, as a message passing process. The computation in Eq.(2.17) and Eq.(2.18) is decomposed into the evalution of two types of messages: those from variables to factors, denoted using M, and those from factors to variables, denoted using M'. Note that $M_{v\to pa(v)}$ is used here in the place of Q_v, and $M'_{u\to v}$ is used in the place of p_v.

message from the variable v to the factor (u, v), and p_v as the message from the factor (u, v) to v.

Consider a generic graphical model, which comprises a set of variables V and a set of factors $\mathcal{F}$. Let $v \in V$ be a variable involved in the factor $a \in \mathcal{F}$, then the message from v to a is

$$M_{v\to a}(x_v) = \prod_{a'\in\mathcal{F}(v)\setminus\{a\}} M'_{a'\to v}(x_v), \tag{2.19}$$

where $\mathcal{F}(v)$ denotes the set of all factors that involve v. The message from a to v is

$$M'_{a\to v}(x_v) = \sum_{x'_{S_a}: x'_v = x_v} f_a(x_{S_a}) \prod_{w \in S_a \setminus \{v\}} M_{w\to a}(x'_w). \tag{2.20}$$

Here, S_a is the set of variables involved in the factor f_a.

For a tree-structured graphical model, after an upward sweep with messages passed from leafs to the root, and then a downward sweep with messages passed from the root to leafs, exact marginals can then be computed as

$$p_v(x_v) \propto \prod_{a \in \mathcal{F}(v)} M'_{a\to v}(x_v). \tag{2.21}$$

The algorithm described above is called ***belief propagation***. Note that belief propagation are often applied to graphical models with loops, which is often referred to as ***loopy belief propagation***. In a loopy belief propagation algorithm, the message passing process may run many cycles. Note that unlike the belief propagation algorithm on trees, which is guaranteed to converge within a finite number of iterations, loopy belief propagation converges only under certain conditions, and even when it converges, the results may not be the exact marginals. Therefore, it is an approximate inference technique.

Junction Trees

Given a graph with cycles, exact inference can be performed based on the *junction tree* representation, where vertices are clustered to form a tree of cliques. A generalized message passing algorithm can be applied to perform exact inference over a junction tree. The computational complexity of this algorithm grows exponentially in the *tree-width*, *i.e.* the size of the maximal clique over all possible triangulations of the underlying graph.

For many graphical models arising in practice, the tree-width may grow with the problem scale, rendering the junction tree algorithm intractable even for a problem of moderate size. In such cases, one may resort to approximate inference techniques.

2.2 Exponential Family Distributions

The notion of exponential family distributions subsume a board range of probabilistic models, which include many classic distributions (*e.g.* Gaussian distributions, multinomial distributions, and Dirichlet distributions, etc) and complex probabilistic models integrating multiple components. In particular, most models developed in this thesis, including the patch-based MRF to describe appearance (see Chapter 3) and the stochastic flow model (see Chapter 4), belong to exponential families.

2.2.1 Basics of Exponential Families

Given a real-valued random vector X taking values in $\mathcal{X}$, and a collection of real valued functions defined on $\mathcal{X}$ as $\boldsymbol{\phi} = (\phi_j)_{j=1}^d$. Then for each $\mathbf{x} \in \mathcal{X}$, $\phi(\mathbf{x}) \triangleq (\phi_j(\mathbf{x}))_{j=1}^d$ is an d-dimensional real vector.

With this notation, the ***exponential family*** associated with ϕ is defined to be a family of parametric distributions, as

$$p(\mathbf{x}) = \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) - A(\boldsymbol{\theta})\right). \tag{2.22}$$

Here, $p(\mathbf{x})$ is the probability density function when X is a continuous variable, or the probability mass function when X is discrete. In addition, $\boldsymbol{\phi}$ is called the ***sufficient statistics***, $\boldsymbol{\theta}$ is called the ***natural parameter*** (also known as the ***canonical parameter***, and A is called the ***log partition function***, which is given by

$$A(\boldsymbol{\theta}) = \int_{\mathcal{X}} \exp(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}))\nu(d\mathbf{x}). \tag{2.23}$$

Here, ν is the base measure (Lebesgue measure for continuous variables and counting measure for discrete variables).

When there exists a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ and a real number C such that

$$\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) = C, \quad a.e.(\text{w.r.t. } \nu),$$

then the formulation above is called a ***overcomplete*** representation. In this case, there exists an entire affine subset of parameters associated with the same distribution. These parameters are unidentifiable from a statistical standpoint.

If the representation is not *overcomplete*, it is called a ***minimal*** representation. If a minimal representation is used, there is a unique parameter associated with each distribution in the family. For each exponential family, one can always find a minimal representation through reparameterization.

Using minimal representation often simplifies theoretical analysis. However, overcomplete representation can be useful and convenient in some practical cases.

Mean Parameterization

The definition above characterizes a exponential family using canonical parameterization. Actually, any exponential family has a dual parameterization, called *mean parameterization*, which describes a distribution in terms of the mean of sufficient statistics. Many statistical computation problems, including marginalization and maximum likelihood estimation, can be considered as converting parameters from one form to the other.

Consider an exponential family distribution p associated with the sufficient statistics $\boldsymbol{\phi} = (\phi_j)_{j=1}^m$. The ***mean parameter*** associated with ϕ_j is defined to be

$$\mu_j = \mathrm{E}_p\left[\phi_j(X)\right] = \int_{\mathcal{X}} \phi_j(\mathbf{x}) p(\mathbf{x}) \nu(d\mathbf{x}).$$

Together, we get a vector of mean parameters $\boldsymbol{\mu} = (\mu_j)_{j=1}^m$. It turns out that under certain conditions, this vector provides an alternative parameterization of the exponential family.

We let $\mathcal{M}$ be the set of all realizable mean vectors, as

$$\mathcal{M} \triangleq \left\{\boldsymbol{\mu} \in \mathbb{R}^d \mid \exists p \text{ s.t. } \mathrm{E}_p[\boldsymbol{\phi}(X)] = \boldsymbol{\mu}\right\}.$$

Proposition 2.3. *For any exponential family, $\mathcal{M}$, the set of all realizable mean*

vectors of sufficient statistics, is convex.

2.2.2 Useful Examples

Next, we consider several probability distributions that will be used in this thesis, and see how they can be analyzed as exponential family distributions.

Gaussian distribution

A multivariate Gaussian distribution over $\mathbb{R}^d$ has a pdf as below

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right). \tag{2.24}$$

With some algebraic manipulation, this can be rewritten as

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp\left(\left((\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})^T\mathbf{x} - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{x}\mathbf{x}^T)\right) - \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + d\log(2\pi) + \log|\boldsymbol{\Sigma}|\right)\right). \tag{2.25}$$

Clearly, this is a exponential family distribution, of which the sufficient statistics, canonical parameters, and log-partition function are respectively given by

$$\boldsymbol{\phi}(\mathbf{x}) = \left(\mathbf{x},\ -\frac{1}{2}\mathbf{x}\mathbf{x}^T\right), \tag{2.26}$$

$$\boldsymbol{\theta} = \left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \boldsymbol{\Sigma}^{-1}\right), \tag{2.27}$$

$$A(\boldsymbol{\theta}) = \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + d\log(2\pi) + \log|\boldsymbol{\Sigma}|\right). \tag{2.28}$$

Discrete distribution

A discrete distribution over a finite set $\mathcal{X}$ is characterized by the probability mass function $f : \mathcal{X} \to [0, 1]$. Let $\mathbf{f} = (f(x))_{x \in xset}$ be an $|\mathcal{X}|$-dimensional vector comprised of all the probability mass values. Then the pmf can be written as

$$p(x|\mathbf{f}) = \mathbf{f}(x) = \mathbf{f}^T I_x \tag{2.29}$$

Here, the sufficient statistics is given by the indicator vector I_x, the natural parameter is $\mathbf{f}$, and the log-partition is always equal to zero. Note that this is an overcomplete formulation, as $\mathbf{1}^T I_x = 1$ always holds.

Dirichlet distribution

A *Dirichlet distribution*, denoted by $\mathrm{Dir}(\boldsymbol{\alpha})$, is a distribution over the probability simplex (*i.e.* the set of all non-negative vectors that sum to 1). The pdf of $\mathrm{Dir}(\boldsymbol{\alpha})$ is given by

$$p(\mathbf{x}|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{i=1}^n \alpha_i\right)}{\prod_{i=1}^n \Gamma(\alpha_i)} \prod_{i=1}^n x_i^{\alpha_i - 1}. \tag{2.30}$$

This can be rewritten in an exponential form as

$$p(\mathbf{x}|\boldsymbol{\alpha}) = (\boldsymbol{\alpha} - \mathbf{1})^T \log(\mathbf{x}) - \left(\sum_{i=1}^n \log \Gamma(\alpha_i) - \log \Gamma\left(\sum_{i=1}^n \alpha_i \right) \right). \tag{2.31}$$

Here, the sufficient statistics is $\log(\mathbf{x})$, the vector formed by entry-wise logarithm, the canonical parameter is $\boldsymbol{\alpha} - 1$, and the log-partition is given by

$$\sum_{i=1}^n \log \Gamma(\alpha_i) - \log \Gamma\left(\sum_{i=1}^n \alpha_i \right).$$

Ising model

The *Ising model*, which originates from statistical physics, has been widely used in computer vision for enforcing smoothness over an indicator map. An Ising model is typically defined on a grid $G = (V, E)$, where each vertex is associated with a binary variable taking values in $\{-1, +1\}$. The joint formulation is defined as an MRF:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z} \prod_{v \in V} \psi_v(x_v) \prod_{(u,v) \in E} \phi_{u,v}(\mathbb{I}(x_u = x_v)). \tag{2.32}$$

Here, Z is a normalization constant.

This can be rewritten in an exponential form as follows. Let

$$\theta_v = (\log \psi_v(1) - \log \psi_v(-1))/2, \quad \text{and} \quad \theta_{uv} = (\log \phi(1) - \log \phi(0))/2,$$

then

$$p(\mathbf{x}|\boldsymbol{\theta}) \propto \exp\left(\sum_{v \in V} \theta_v x_v + \sum_{(u,v) \in E} \theta_{uv} \mathbb{I}(x_u = x_v)\right). \tag{2.33}$$

The re-parameterized formulation above clearly suggests that the Ising model is an exponential family distribution.

2.2.3 The Log-partition Function

The log-partition function $A(\boldsymbol{\theta})$ in itself is an object of particular interest. As stated by the following proposition, the derivatives of A *w.r.t.* the canonical parameter are closely related to the mean parameters.

Proposition 2.4. *Consider an exponential family as given in Eq.(2.22), the log-partition function A has*

$$\nabla_{\boldsymbol{\theta}} A(\boldsymbol{\theta}) = \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(X)], \tag{2.34}$$

and

$$\nabla^2_{\boldsymbol{\theta}} A(\boldsymbol{\theta}) = \mathrm{Cov}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(X)] = \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(X)\boldsymbol{\phi}(X)^T] - \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(X)]\mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(X)]^T. \tag{2.35}$$

Here, $\mathrm{E}_{\boldsymbol{\theta}}[\cdot]$ indicates the expectation w.r.t. the distribution in this family with canonical parameter $\boldsymbol{\theta}$.

As covariance matrix is always semi-definite, Eq.(2.35) immediately leads to the following corollary.

Corollary 2.1. *The log-partition function A is a convex function for any exponential family. Moreover, if A is associated with a minimal representation, then A is strictly convex.*

From Eq.(2.34), we can see that ∇A defines a mapping from the set of all valid canonical parameters, denoted by Θ, to $\mathcal{M}$, the set of realizable mean vectors. For this mapping, we have the following theorem

Theorem 2.2. *For an exponential family with minimal representation, the gradient mapping $\nabla A : \Theta \to \mathcal{M}$ is one-to-one, and onto the interior of $\mathcal{M}$. Conversely, if ∇A is one-to-one, then the associated exponential representation if minimal.*

2.2.4 Conjugate Duality

The conjugate dual of the log-partition function A, denoted by A^*, is defined to be

$$A^*(\boldsymbol{\mu}) \triangleq \sup_{\boldsymbol{\theta} \in \Theta} \left\{ \boldsymbol{\mu}^T \boldsymbol{\theta} - A(\boldsymbol{\theta}) \right\}. \tag{2.36}$$

Here, $\boldsymbol{\mu}$ is called the *dual variable.* The dual function is closely related to the entropy, as formally stated by the theorem below.

Theorem 2.3. *Given an exponential family as in Eq.(2.22). Let $\boldsymbol{\theta}$ be a canonical parameter satisfying the* ***dual matching condition*** *as*

$$\mathrm{E}_{\boldsymbol{\theta}} \left[\boldsymbol{\phi}(X) \right] = \nabla A(\boldsymbol{\theta}) = \boldsymbol{\mu}. \tag{2.37}$$

Then the conjugate dual function A^* has

$$A^*(\boldsymbol{\mu}) = \begin{cases} -H(p_{\boldsymbol{\theta}}) & (\boldsymbol{\mu} \in \mathcal{M}), \\ +\infty & (\textit{otherwise}). \end{cases} \tag{2.38}$$

Here, $H(p_{\boldsymbol{\theta}})$ is the entropy of a distribution p in this family, with parameter $\boldsymbol{\theta}$.

Note that whereas there can be multiple canonical parameters $\boldsymbol{\theta}$ satisfying the dual matching condition when the exponential representation is overcomplete, Eq.(2.38) is well-defined, as all $\boldsymbol{\theta}$ corresponding to the same $\boldsymbol{\mu}$ gives rise to the same distribution, and thus the same entropy value.

Taking advantage of the duality, we further have

Theorem 2.4. *Let A^* be the dual function of a log-partition function A, then*

$$A(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu} \in \mathcal{M}} \left\{ \boldsymbol{\theta}^T \boldsymbol{\mu} - A^*(\boldsymbol{\mu}) \right\}. \tag{2.39}$$

For all $\boldsymbol{\theta} \in \Theta$, the supremum is attained uniquely at $\boldsymbol{\mu}$ that satisfies the dual matching condition.

2.3 Model Estimation and Variational Inference

This section considers the problem of estimating the parameters of an exponential family model from observed data. Two approaches are discussed here: the *frequentist approach*, which pursues the maximum likelihood estimate (MLE) of the parameters, and the *Bayesian approach*, which considers the parameters as random variables generated from a prior distribution.

Generally, in the simplest cases where variables generated from a model is completely observed, the maximum likelihood estimation reduces to the problem of moment matching. However, when variables are partially observed, one may resort to Expectation-Maximization (EM) algorithm, which provides a general approach to solving MLE. The E-steps in an EM algorithm requires evaluation of the mean parameters, which, for some complex model, can be computationally intractable. Variational inference based on mean field approximation is a generic methodology to address this difficulty.

2.3.1 Maximum Likelihood Estimation

Consider a probability distribution $p(\mathbf{x}|\boldsymbol{\theta})$, whose parameter $\boldsymbol{\theta}$ is unknown and to be estimated. Let $X_1, \ldots, X_n$ be n independent and identically distributed (i.i.d.) variables, each generated from p.

Estimation of completely observed models

We first consider the simplest situation, where the values of $X_1, \ldots, X_n$ are completely observed (or known), which are denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Then the ***maximum likelihood estimate (MLE)***, denoted by $\hat{\boldsymbol{\theta}}$, is defined to be

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta}).$$

Taking the logarithm of the likelihood values turns the product into sum, which, in most cases, would greatly simplify the evaluation. Thus, $\hat{\boldsymbol{\theta}}$ can be reformulated equivalently as

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\theta}). \tag{2.40}$$

Let $p(\mathbf{x}|\boldsymbol{\theta})$ be an exponential family, as

$$p(\mathbf{x}|\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) - A(\boldsymbol{\theta})), \tag{2.41}$$

where $\boldsymbol{\theta}$ is the vector canonical parameters. Then Eq.(2.40) further reduces to

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ \boldsymbol{\theta}^T \bar{\boldsymbol{\mu}} - A(\boldsymbol{\theta}), \tag{2.42}$$

where $\bar{\boldsymbol{\mu}}$ is the *empirical mean* of the sufficient statistics over the observed data, which is given by

$$\bar{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\phi}(\mathbf{x}_i). \tag{2.43}$$

According to Theorem 2.3, the optimum of Eq.(2.42) is attained, when

$$\mathrm{E}_{\boldsymbol{\theta}} [\phi(\mathbf{x})] = \bar{\boldsymbol{\mu}}. \tag{2.44}$$

Hence, the problem of maximum likelihood estimation for an exponential family reduces to a moment matching problem, which pursues a parameter with which the expectation of sufficient statistics matches the empirical mean.

Except for some simple distributions, where analytic solution exists, solving MLE generally requires the use of numerical optimization methods (*e.g.* gradient descend). Note that since $A(\boldsymbol{\theta})$ is convex, the optimization problem in Eq.(2.42) is convex.

Estimation of Models with Latent Variables

A more challenging problem arises when parts of the variables are not observed directly. For convenience of discussion, for a partially observed model, we denote the vector of variables generated from the model as (X, Y), which is comprised of an observed part X and a *latent* part Y. In addition, each assignment to (X, Y) is denoted by $(\mathbf{x}, \mathbf{y})$ with $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$.

Suppose the model is an exponential family. The joint formulation can then be written as

$$p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) = \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) - A(\boldsymbol{\theta})\right). \tag{2.45}$$

Given an observation $X = \mathbf{x}$, the posterior distribution of Y is

$$p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})\right)}{\int_{\mathcal{Y}} \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})\right) \nu(d\mathbf{y})} \tag{2.46}$$

Clearly, this remains an exponential family distribution, of which the associated sufficient statistics is $\boldsymbol{\phi}_{\mathbf{x}}(\mathbf{y}) \triangleq \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$. For this distribution, the log partition function $A_{\mathbf{x}}(\boldsymbol{\theta})$ is given by

$$A_{\mathbf{x}}(\boldsymbol{\theta}) = \int_{\mathcal{Y}} \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})\right) \nu(d\mathbf{y}). \tag{2.47}$$

Note here that the integration is over $\mathcal{Y}$, and $A_{\mathbf{x}}$ depends on $\mathbf{x}$.

Given partial observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$ is defined to be

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\theta}). \tag{2.48}$$

Here, $p(\mathbf{x}_i|\boldsymbol{\theta})$ is the marginal likelihood of $\mathbf{x}_i$, given by

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \int_{\mathcal{Y}} p(\mathbf{x}_i, \mathbf{y}|\boldsymbol{\theta}) \nu(d\mathbf{y}). \tag{2.49}$$

Under the exponential family formulation in Eq.(2.45), it can be rewritten as

$$\begin{aligned} p(\mathbf{x}_i|\boldsymbol{\theta}) &= \int_{\mathcal{Y}} \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) - A(\boldsymbol{\theta})\right) \nu(d\mathbf{y}) \\ &= \exp(-A(\boldsymbol{\theta})) \int_{\mathcal{Y}} \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y})\right) \nu(d\mathbf{y}). \end{aligned} \tag{2.50}$$

Combining this with Eq.(2.47) results in

$$\log p(\mathbf{x}_i|\boldsymbol{\theta}) = A_{\mathbf{x}}(\boldsymbol{\theta}) - A(\boldsymbol{\theta}). \tag{2.51}$$

We can see that the partial log-likelihood can be characterized as the difference between the log partition of $p(\mathbf{y}|\mathbf{x}_i; \boldsymbol{\theta})$ and that of $p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})$.

2.3.2 Expectation Maximization

Consider the partially observed model as formulated above. Generally, evaluation of $A_{\mathbf{x}}(\boldsymbol{\theta})$ requires integration over $\mathcal{Y}$, which is often intractable. This difficulty can be addressed using an variational lower bound to approximate $A_{\mathbf{x}}$.

Variational Lower Bound

Let $\mathcal{M}_{\mathbf{x}}$ denote the set of all realizable mean vectors for the exponential family $p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta})$, as

$$\mathcal{M}_{\mathbf{x}} = \left\{ \exists p : \ \boldsymbol{\mu} \in \mathbb{R}^d \mid \boldsymbol{\mu} = \mathrm{E}_p \left[\boldsymbol{\phi}(\mathbf{x}, Y) \right] \right\}. \tag{2.52}$$

Then, according to Theorem 2.4, we obtain the variational representation of $A_{\mathbf{x}}$ as

$$A_{\mathbf{x}}(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu}_x \in \mathcal{M}_x} \left\{ \boldsymbol{\theta}^T \boldsymbol{\mu}_x - A^*_{\mathbf{x}}(\boldsymbol{\mu}_x) \right\}, \tag{2.53}$$

with

$$A^*_{\mathbf{x}}(\boldsymbol{\mu}_x) = \sup_{\boldsymbol{\theta} \in \Theta} \left\{ \boldsymbol{\theta}^T \boldsymbol{\mu}_x - A_{\mathbf{x}}(\boldsymbol{\theta}) \right\}. \tag{2.54}$$

Then the partial log-likelihood in Eq.(2.51) can be bounded from below, as

$$\log p(\mathbf{x}_i|\boldsymbol{\theta}) \geq \boldsymbol{\theta}^T \boldsymbol{\mu}_x - A^*_{\mathbf{x}}(\boldsymbol{\mu}_x) - A(\boldsymbol{\theta}). \tag{2.55}$$

This inequality holds for any $\boldsymbol{\mu}_x \in \mathcal{M}_x$. If and only if the dual matching condition is satisfied, as

$$\boldsymbol{\mu}_x = \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(\mathbf{x}, Y)] = \nabla_{\boldsymbol{\theta}} A_{\mathbf{x}}(\boldsymbol{\theta}), \tag{2.56}$$

the inequality becomes an equality.

Expectation Maximization Algorithm

With the analysis above, we can derive a variational formulation, where the objective function is given by

$$L(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_n, \boldsymbol{\theta}) = \sum_{i=1}^{n} L_i(\boldsymbol{\mu}_i, \boldsymbol{\theta}) = \sum_{i=1}^{n} \left(\boldsymbol{\theta}^T \boldsymbol{\mu}_i - A^*_{\mathbf{x}_i}(\boldsymbol{\mu}_i) - A(\boldsymbol{\theta}) \right). \tag{2.57}$$

According to Eq.(2.56), this objective function provides a lower bound of $\sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\theta})$.

The ***Expectation-Maximization (EM)*** algorithm is a coordinate ascent scheme to optimize this variational lower-bound, which alternates between two updating steps, as below.

1. ***E-step***: update each dual vector $\boldsymbol{\mu}_i$ with the model parameter $\boldsymbol{\theta}$ fixed, as

$$\boldsymbol{\mu}_i^{(t+1)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\mu} \in \mathcal{M}_{\mathbf{x}_i}} L_i(\boldsymbol{\mu}, \boldsymbol{\theta}^{(t)}). \tag{2.58}$$

   This problem can be further reduced to

$$\boldsymbol{\mu}_i^{(t+1)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\mu} \in \mathcal{M}_{\mathbf{x}_i}} (\boldsymbol{\theta}^{(t)})^T \boldsymbol{\mu} - A^*_{\mathbf{x}_i}(\boldsymbol{\mu}) \tag{2.59}$$

   The optimum is attained when $\boldsymbol{\mu} = \mathrm{E}_{\boldsymbol{\theta}^{(t)}}[\phi(\mathbf{x}_i, Y)]$. Hence this step is to solve

the mean of $\phi(\mathbf{x}_i, Y)$ *w.r.t.* $\boldsymbol{\theta}^{(t)}$, as

$$\boldsymbol{\mu}_i^{(t+1)} \leftarrow \mathrm{E}_{\boldsymbol{\theta}^{(t)}}\left[\phi(\mathbf{x}_i, Y)\right]. \tag{2.60}$$

2. ***M-step***: update the parameter $\boldsymbol{\theta}$ with all dual vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_n$ fixed, as

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \sum_{i=1}^{n} L_i(\boldsymbol{\mu}, \boldsymbol{\theta}_i^{(t+1)}). \tag{2.61}$$

This can be further written as

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \sum_{i=1}^{n} \left(\boldsymbol{\theta}^T \boldsymbol{\mu}_i^{(t+1)} - A(\boldsymbol{\theta})\right). \tag{2.62}$$

Clearly, this is equivalent to the maximum likelihood estimate over a fully observed model, with the sufficient statistics of the observation data given by $\boldsymbol{\mu}_1^{(t+1)}, \ldots, \boldsymbol{\mu}_n^{(t+1)}$. In particular, the optimum is attained when

$$\mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(X, Y)] = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\mu}_i^{(t+1)}. \tag{2.63}$$

It is worth noting that at the end of each E-step, the gap between the sum of partial log likelihood and the variational lower bound will be closed, as

$$L_i(\boldsymbol{\mu}_i^{(t+1)}, \boldsymbol{\theta}^{(t)}) = \log p(\mathbf{x}_i|\boldsymbol{\theta}^{(t)}). \tag{2.64}$$

Hence, as the algorithm proceeds, the variational lower bound increases, and thus the joint partial likelihood also increases.

A key computational challenge of the E-M algorithm is the evaluation of $E_{\boldsymbol{\theta}}\left[\phi(\mathbf{x}_i, Y)\right]$ in E-steps. Specifically, this mean is given by

$$E_{\boldsymbol{\theta}}\left[\phi(\mathbf{x}_i, Y)\right] = \int_{\mathcal{Y}} \phi(\mathbf{x}_i, \mathbf{y}) p(\mathbf{y}|\mathbf{x}_i; \boldsymbol{\theta}) \nu(d\mathbf{y}). \tag{2.65}$$

Except for some special models, computation of this integral (or sum) is usually in-

tractable. Variational inference, as we shall see next, provides a generic methodology to address this difficulty.

2.3.3 Mean Field and Variational Inference

Evaluation of the mean of sufficient statistics arises not only in uvarious inference problems but also in model estimation (*e.g.* EM algorithm requires evaluation of conditional mean vectors in E-steps). In general, such evaluation involves integration (or sum) over some space, which can be intractable in complex models. In what follows, we describe a generic approach – *variational inference* – to address this problem, which relies on *mean field approximation*, a tool originating from statistical physics.

Tractable family

We begin the exposition by introducing the notion of *tractable family*. Given a graphical model based on a graph $G = (V, E)$, a ***tractable subgraph*** is a subgraph F of G over which it is feasible to perform exact inference. For example, F can be a completely disconnected graph (*i.e.* a graph without edges), a chain, or a tree. Then the family of probabilistic distributions that can factorize according to F is a sub-family of those defined on G.

Consider an exponential family with a set of sufficient statistics $\boldsymbol{\phi} = (\phi_j)_{j=1}^m$, each associated with a clique of G. Only a subset of them is associated with the cliques of the subgraph F, which we denote by $I_F \subset \{1, \dots m\}$. Then, the canonical parameters of the distribution in the sub-family defined on F is a subset of Θ, as

$$\Theta(F) \triangleq \{\boldsymbol{\theta} \in \Theta \mid \theta_j = 0, \ \forall j \in \{1, \dots, m\} \backslash F\}. \tag{2.66}$$

A simple example may help to illustrate these notations. Consider an Ising model defined on a graph $G = (V, E)$, as below

$$p(\mathbf{x}|\boldsymbol{\theta}) = \exp\left(\sum_{v \in V} \theta_v x_v + \sum_{\{u,v\} \in E} \theta_{uv} \mathbb{I}(x_u = x_v) - A(\boldsymbol{\theta})\right). \tag{2.67}$$

Here, the parameter space Θ is a subset of $\mathbb{R}^{|V|+|E|}$. Given a disconnected subgraph $F = (V, \emptyset)$. The pairwise sufficient statistics $\mathbb{I}(x_u = x_v)$ is no longer associated with the cliques of F, and therefore I_F only contains the indices of the single-variable potentials. As a result, the restricted parameter space for F is

$$\Theta(F) = \{\boldsymbol{\theta} \in \Theta | \theta_{uv} = 0, \ \forall \{u, v\} \in E\}. \tag{2.68}$$

Let $\mathcal{M}$ be the set of all realizable mean vectors for the graphical model defined on G. For the sub-family defined on F, the set is realizable mean vectors is denoted by $\mathcal{M}_F$, as

$$\mathcal{M}(F) \triangleq \{\boldsymbol{\mu} \in \mathbb{R}^m \mid \exists \boldsymbol{\theta} \in \Omega(F) : \ \boldsymbol{\mu} = E_{\boldsymbol{\theta}}[\boldsymbol{\phi}(\mathbf{x})]\}. \tag{2.69}$$

Clearly, $\mathcal{M}(F)$ is a convex subset of $\mathcal{M}$.

Mean field approximation

The following proposition is the theoretical basis for mean field methods.

Proposition 2.5. *Let A be a log-partition function associated with an exponential family, then for each $\boldsymbol{\theta} \in \Theta$,*

$$A(\boldsymbol{\theta}) \geq \boldsymbol{\theta}^T \boldsymbol{\mu} - A^*(\boldsymbol{\mu}), \ \forall \boldsymbol{\mu} \in \mathcal{M}. \tag{2.70}$$

Here, the dual function A^* is defined by

$$A^*(\boldsymbol{\mu}) := \sup_{\boldsymbol{\theta} \in \Theta} \left\{\boldsymbol{\theta}^T \boldsymbol{\mu} - A(\boldsymbol{\theta})\right\}. \tag{2.71}$$

The equality holds if and only if $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ satisfy the $\boldsymbol{\mu} = \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(\mathbf{x})]$.

Here, Eq.(2.70) provides a lower bound of $A(\boldsymbol{\theta})$. However, A^* is often intractable to evaluate general. In such cases, it is infeasible to compute this lower bound. The ***mean field method*** approxmiates this lower bound by restricting $\boldsymbol{\mu}$ to the tractable subset $\mathcal{M}(F)$[1].

[1]As we assume it is feasible to perform exact inference on F, evaluating $\boldsymbol{\mu} \in \mathcal{M}(F)$ is feasible.

Let A_F^* be the restriction of A^* to $\mathcal{M}(F)$, which is tractable to compute. Then $\boldsymbol{\theta}^T\boldsymbol{\mu} - A_F^*(\boldsymbol{\mu})$ for each $\boldsymbol{\mu} \in \mathcal{M}(F)$ provides a tractable lower bound of $A(\boldsymbol{\theta})$. Among all these tractable lower bounds, the best one (*i.e.* the tightest one) is

$$\sup_{\boldsymbol{\mu}\in\mathcal{M}(F)} \left\{\boldsymbol{\theta}^T\boldsymbol{\mu} - A_F^*(\boldsymbol{\mu})\right\}. \tag{2.72}$$

This is called the ***mean field approximation*** of $A(\boldsymbol{\theta})$. Solving this optimization problem, we get

$$\hat{\boldsymbol{\mu}}_F = \operatorname*{argmax}_{\boldsymbol{\mu}\in\mathcal{M}(F)} \boldsymbol{\theta}^T\boldsymbol{\mu} - A_F^*(\boldsymbol{\mu}) \tag{2.73}$$

Here, $\hat{\boldsymbol{\mu}}_F$ is called the ***mean field approximation*** of $\mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(\mathbf{x})]$.

The *mean field approximation* can also be derived and interpreted in a different way, in terms of minimizing the *Kullback-Leibler divergence.*

Recall that an exponential family distribution can be characterized using either the canonical parameters or the mean parameters. Hence, a mean vector $\boldsymbol{\mu} \in \mathcal{M}$ uniquely corresponds to a distribution in the family. Consider a distribution $p_{\boldsymbol{\theta}}$ with canonical parameter $\boldsymbol{\theta}$, and a distribution $q_{\boldsymbol{\mu}}$ with mean vector $\boldsymbol{\mu}$. The *Kullback-Leibler divergence* between them is

$$\begin{aligned} D_{KL}(q_{\boldsymbol{\mu}}||p_{\boldsymbol{\theta}}) &= \mathrm{E}_{q_{\boldsymbol{\mu}}}\left[\log\frac{q_{\boldsymbol{\mu}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}\right] \\ &= \mathrm{E}_{q_{\boldsymbol{\mu}}}\left((\boldsymbol{\theta}_\mu^T\boldsymbol{\phi}(\mathbf{x}) - A(\boldsymbol{\theta}_\mu)) - (\boldsymbol{\theta}^T\boldsymbol{\phi}(\mathbf{x}) - A(\boldsymbol{\theta})\right) \\ &= (\boldsymbol{\theta}_\mu - \boldsymbol{\theta})^T\boldsymbol{\mu} + A(\boldsymbol{\theta}) - A(\boldsymbol{\theta}_\mu). \end{aligned} \tag{2.74}$$

Here, $\boldsymbol{\theta}_\mu$ denotes a canonical parameter dually coupled with $\boldsymbol{\mu}$ (*i.e.* satisfying the dual matching condition) and we utilize the fact $\mathrm{E}_{q_{\boldsymbol{\mu}}}[\boldsymbol{\phi}(\mathbf{x})] = \boldsymbol{\mu}$. Moreover, as $\boldsymbol{\theta}_\mu$ is dually coupled with $\boldsymbol{\mu}$, we have

$$\boldsymbol{\theta}_\mu^T\boldsymbol{\mu} - A(\boldsymbol{\theta}_\mu) = A^*(\boldsymbol{\mu}). \tag{2.75}$$

This immediately follows that

$$D_{KL}(q_{\boldsymbol{\mu}}||p_{\boldsymbol{\theta}}) = A(\boldsymbol{\theta}) + A^*(\boldsymbol{\mu}) - \boldsymbol{\theta}^T\boldsymbol{\mu}. \tag{2.76}$$

Restricting $\boldsymbol{\mu}$ to the tractable subset $\mathcal{M}(F)$, we have

$$D_{KL}(q_{\boldsymbol{\mu}}||p_{\boldsymbol{\theta}}) = A(\boldsymbol{\theta}) + A_F^*(\boldsymbol{\mu}) - \boldsymbol{\theta}^T\boldsymbol{\mu}, \quad \forall \boldsymbol{\mu} \in \mathcal{M}(F). \tag{2.77}$$

Thus,

$$\hat{\boldsymbol{\mu}}_F = \underset{\boldsymbol{\mu}\in\mathcal{M}(F)}{\operatorname{argmax}} \left(\boldsymbol{\theta}^T\boldsymbol{\mu} - A_F^*(\boldsymbol{\mu})\right) = \underset{\boldsymbol{\mu}\in\mathcal{M}(F)}{\operatorname{argmin}} D_{KL}(q_{\boldsymbol{\mu}}||p_{\boldsymbol{\theta}}). \tag{2.78}$$

This result suggests that the *mean field approximation* to $\mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(\mathbf{x})]$ is the mean vector $\boldsymbol{\mu}$ in $\mathcal{M}(F)$ that minimizes the Kullback-Leibler divergence between $q_{\boldsymbol{\mu}}$ and $p_{\boldsymbol{\theta}}$.

Variational E-M

Come back to the E-M algorithm for maximum likelihood estimation. When the mean vectors in E-steps are difficult or even intractable to evaluate, one may resort to variational inference techiques to approximate them. This variant of the EM algorithm is often referred to as ***variational EM***.

In variational EM, the objective function remains the same as Eq.(2.57), except that the mean vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_n$ are restricted to a tractable subset $\mathcal{M}(F)$. Then, the problem in E-steps is given by

$$\boldsymbol{\mu}_i^{(t+1)} \leftarrow \underset{\boldsymbol{\mu}\in\mathcal{M}_{\mathbf{x}_i}(F)}{\operatorname{argmax}} \left((\boldsymbol{\theta}^{(t)})^T\boldsymbol{\mu} - A_{\mathbf{x}_i}^*(\boldsymbol{\mu})\right) = \underset{\boldsymbol{\mu}\in\mathcal{M}_{\mathbf{x}_i}(F)}{\operatorname{argmin}} D_{KL}(q_{\boldsymbol{\mu}}||p_{\boldsymbol{\theta}^{(t)}}). \tag{2.79}$$

As the optimum is attained, the objective value $L_i(\boldsymbol{\mu}_i^{(t+1)}, \boldsymbol{\theta}^{(t)})$, has

$$L_i(\boldsymbol{\mu}_i^{(t+1)}, \boldsymbol{\theta}^{(t)}) = \log p(\mathbf{x}_i|\boldsymbol{\theta}^{(t)}) - D_{KL}(q_{\boldsymbol{\mu}^{(t+1)}}||p_{\boldsymbol{\theta}^{(t)}}). \tag{2.80}$$

When $\mathcal{M}_F = \mathcal{M}$, the variational EM algorithm degenerates to the standard EM, where $D_{KL}(q_{\boldsymbol{\mu}^{(t+1)}}||p_{\boldsymbol{\theta}^{(t)}}) = 0$, and the variational lower bound is tight.

2.4 Monte Carlo Sampling

Evaluation of expectation often arises as central steps in probabilistic inference and model estimation, which is often a very challenging task as it requires integration over a huge space in many practical cases. There are many approaches to address this problem, typically relying on approximations. A representative method is variational inference based on mean field approximation. This section describes another category of methods, namely *Monte Carlo sampling*, which performs inference by drawing samples from the desired distribution.

2.4.1 Monte Carlo Integration

Consider the problem of evaluating the expectation of a real-valued random vector $X \sim p$, as

$$\mathrm{E}_p[X] = \int_{\mathcal{X}} \mathbf{x} p(\mathbf{x}) \nu(\mathbf{x}). \tag{2.81}$$

When this integration is difficult to compute, we can approximate it by independently drawing a large number of samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from p and computing the sample mean, as

$$\mathrm{E}_p[X] \simeq \hat{x} \triangleq \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i. \tag{2.82}$$

More generally, we can approximate $h(X)$ as

$$\mathrm{E}_p[h(X)] \simeq \hat{h} \triangleq \frac{1}{n} \sum_{i=1}^{n} h(\mathbf{x}_i) \tag{2.83}$$

Here, $\mathbf{h} : \mathcal{X} \to \mathbb{R}$ is an arbitrary real-valued function. The estimator $\hat{h}$ is unbiased, meaning

$$\mathrm{E}_p[\hat{h}] = \mathrm{E}_p[h(\mathbf{x})].$$

Moreover, according to the *Law of Large Numbers*, $\hat{h}$ almost surely converges to the expectation $\mathrm{E}_p[h(\mathbf{x})]$, as $n \to \infty$. The variance of $\hat{h}$ is given by

$$\mathrm{Var}(\hat{h}) = \frac{1}{n} \mathrm{E}_p \left[(\hat{h} - \mathrm{E}_p[h])^2 \right]. \tag{2.84}$$

2.4.2 Importance Sampling

In the cases where p is difficult to sample from, one can draw samples from another distribution q and reweight the samples. Specifically, given $X \sim p$ and a real-valued function h defined on $\mathcal{X}$, we have

$$\mathrm{E}_p[h(\mathbf{x})] = \int_{\mathcal{X}} h(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} p(\mathbf{x}) d\mathbf{x} = \mathrm{E}_q \left[\frac{p(\mathbf{x})}{q(\mathbf{x})} h(\mathbf{x}) \right]. \tag{2.85}$$

Thus, we can draw independent samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from q, and approximate $\mathrm{E}_p[h]$ as

$$\mathrm{E}_p[h(\mathbf{x})] \simeq \hat{h}_q = \frac{1}{n} \sum_{i=1}^{n} \frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)} h(\mathbf{x}_i). \tag{2.86}$$

This way of doing Monte Carlo integration is called ***importance sampling***. Here, q can be arbitrary distribution that satisfies

$$q(\mathbf{x}) = 0 \;\Rightarrow\; p(\mathbf{x}) = 0. \tag{2.87}$$

Generally, it is desirable to choose q such that $q(\mathbf{x})$ is roughly proportional to $p(\mathbf{x})h(\mathbf{x})$, which would lead to small variance of the estimator.

In practice, p and q are often formulated in the following form

$$p(\mathbf{x}) = \frac{1}{Z_p} g_p(\mathbf{x}), \quad \text{and} \quad q(\mathbf{x}) = \frac{1}{Z_q} g_q(\mathbf{x}). \tag{2.88}$$

Here, Z_p and Z_q are normalization constants, which may be difficult to compute. In such cases, one can approximate $\mathrm{E}_p[h]$ using $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim q$ as follows

$$\mathrm{E}_p\left[h(\mathbf{x})\right] = \frac{\sum_{i=1}^{n} w_i h(\mathbf{x}_i)}{\sum_{i=1}^{n} w_i}, \quad \text{with } w_i = \frac{g_p(\mathbf{x}_i)}{g_q(\mathbf{x}_i)}. \tag{2.89}$$

Here, w_i is often called the *importance weight* of $\mathbf{x}_i$.

2.4.3 Markov Chain Monte Carlo

Direct sampling from a high dimensional space is usually very difficult. *Markov Chain Monte Carlo (MCMC)* provides a very general and powerful methodology to address this problem, which makes it feasible to sample from a broad class of complex distributions.

Ergodic Markov chain

Markov Chain Monte Carlo is developed based on the *ergodicity* of Markov chains. A ***Markov chain*** is defined to be a sequence of random variables $X_1, X_2, \ldots$, such that

$$p(X_{t+1}|x_t, x_{t-1}, \ldots, \mathbf{x}_1) = p(X_{t+1}|x_t), \quad t = 1, 2, \ldots \tag{2.90}$$

If the transition probability $p(x_{t+1}|x_t)$ is time invariant, the Markov chain is said to be ***homogeneous***. A *homogeneous Markov chain* is completely characterized by a transition probability matrix $\mathbf{T}$, with which, we have $p(x_{t+1}|x_t) = \mathbf{T}(x_t, x_{t+1})$.

A distribution p over $\mathcal{X}$ is called a ***stationary distribution*** (or invariance distribution) *w.r.t.* the Markov chain with transition probability matrix $\mathbf{T}$, if

$$p(x) = \sum_{x' \in \mathcal{X}} p(x')\mathbf{T}(x', x). \tag{2.91}$$

Proposition 2.6. *Given a Markov chain with transition probability matrix* $\mathbf{T}$*, let* p *be a distribution over* $\mathcal{X}$*, then* p *is a stationary distribution w.r.t. the given Markov chain, if*

$$p(x)\mathbf{T}(x, x') = p(x')\mathbf{T}(x', x). \tag{2.92}$$

Here, the condition given in Eq.(2.92) is called ***detailed balance***, which is a sufficient condition that ensures p is a stationary distribution *w.r.t.* $\mathbf{T}$.

The primary goal here is to sample from a given distribution by constructing a Markov chain. This is related to an important property of a Markov chain – *ergodicity.*

Definition 2.3 (Ergodicity). *A Markov chain over a finite state space is said to be* ***ergodic****, if the following conditions holds:*

1. *The Markov chain is* ***irreducible****, meaning every state is accessible from every other state.*

2. *Every state s is* ***positive recurrent****, meaning that starting from s, the chain will return to s almost surely, and the expected time to return to s is finite.*

3. *Every state s is* ***aperiodic****, meaning that there exists a time t_s, such that a chain starting from s may return to s at any time step after t_s.*

The following theorem is a fundamental result of Markov chain theory, which also serves as the theoretical basis of MCMC.

Theorem 2.5. *There exists a unique stationary distribution π for an ergodic Markov chain, which is called the* ***equilibrium distribution****. Moreover, in spite of the initial distribution of X_1, the sequence $X_1, X_2, \ldots$ converges in distribution to π.*

While the introduction above focues on Markov chains over finite state space, similar analysis can be applied to Markov processes over other measurable spaces.

The Metropolis-Hastings algorithm

The basic idea of *Markov Chain Monte Carlo (MCMC)* is to simulate an ergodic Markov chain whose equilibrium distribution p is the distribution one wants to draw samples from. After running the chain for a sufficiently long time, the distribution of the samples generated from the chain matches the desired distribution p.

The famous *Metropolis-Hastings algorithm (M-H)* is a sampling method that follows this idea, which is a generalization of the basic Metropolis algorithm. The M-H algorithm requires a proposal distribution q, which is used to generate a new sample conditioned on the current one. The detailed procedure of this algorithm is given in Algorithm 1.

This algorithm is actually simulating a Markov chain whose equilibrium distribution is p. This can be easily shown by proving the *detailed balance condition* as follows

$$p(x)q(x|x')A(x',x) = \min\{p(x)q(x|x'), p(x)q(x|x')\} = p(x')q(x'|x)A(x,x'). \quad (2.94)$$

Algorithm 1 Metropolis-Hastings algorithm

Start with an arbitrary initial state $x_0 \in \mathcal{X}$.
for $t = 1, 2, \ldots$ **do**
Propose a new sample x' from the proposal distribution $q(x'|x_{t-1})$;
Calculate the acceptance ratio as

$$a_t = A(x', x_{t-1}) = \min\left(1, \frac{p(x')q(x_{t-1}|x')}{p(x_{t-1})q(x'|x_{t-1})}\right). \tag{2.93}$$

if $a_t = 1$ **then**
Accept x', and set $x_{t+1} = x'$.
else
Accept x' with probability a_t. Specifically, draw $u \sim U([0, 1])$, and set

$$x_{t+1} = \begin{cases} x' & (u \leq a_t), \\ x_t & (u > a_t). \end{cases}$$

end if
end for
Stops when enough samples have been acquired.

In general, an MCMC algorithm need to be run for a long time (often referred to as the ***burn-in*** stage) before the samples are collected. This is to ensure that the chain is close enough to the equilibrium distribution. Moreover, consecutive samples are correlated. In practice, one can only take every n samples in Monte Carlo integration, to ensure that the correlation between used samples are negligible. Here, the interval n should be chosen depending on how fast the correlation attenuates.

Theoretically, one can choose arbitrary distribution q as the proposal distribution, as long as one can travel from one state to any other state within finite number of steps via the proposal kernel. That being said, the specific choice of q has remarkable influence on the efficiency of the algorithm. In general, a good choice allows large moves to escape local traps while minizing the rejection rate. Devising a good proposal distribution is often more an art than a technique.

2.4.4 Gibbs Sampling

Gibbs sampling is a simple and widely used MCMC algorithm and can be considered as a special case of the Metropolis-Hastings algorithm.

Suppose we are to sample from a joint distribution $p(\mathbf{x}) = p(x_1, \ldots, x_m)$. Starting from an atrbitrary vector, the ***Gibbs sampling*** algorithm updates one component of the vector at a step as follows: (1) choose a particular component (say x_i), and (2) re-draw the value of x_i from the conditional distribution $p(x_i|\mathbf{x}_{\setminus i})$. At each cycle, the algorithm updates all variables following a prescribed or random order.

The Gibbs sampling procedure gives rise to a Markov chain, as the distribution of the state produced by each step is solely determined by the previous state. Moreover, it is not difficult to see that p is invariant *w.r.t.* this Markov chain. This can be shown by the following argument. Suppose the joint distribution over all variables at current step is p. At next step, we redraw x_i from $p(x_i|\mathbf{x}_{\setminus i})$. Note that the marginal distribution of $\mathbf{x}_{\setminus i}$ remains the same. Hence, the joint distribution remains p.

The Gibbs sampling algorithm can also be viewed as a special case of the Metropolis-Hastings algorithm, where the proposal distribution for the step to update x_i is

$$q((x_i', \mathbf{x}_{\setminus i})|(x_i, \mathbf{x}_{\setminus i})) = p(x_i'|\mathbf{x}_{\setminus i}). \tag{2.95}$$

It is not difficult to verify with such a proposal, the acceptance ratio is always 1, implying that the proposed transition is always accepted.

The standard Gibbs sampler, though simple, is usually very inefficient, as consecutive samples generated from the chain is highly correlated, especially for high-dimensional sample spaces. To improve the performance of the Gibbs sampler, two widely used strategies are

1. ***Blocked Gibbs sampling***: group variables into blocks, and update an entire block at a step, conditioned on other blocks.

2. ***Collapsed Gibbs sampling***: integrates out one or more variables when sampling from other variables.

Chapter 3

The Appearance Model

One important module of the scene modeling framework is the appearance model. Technically, an image is a two dimensional matrix of numbers, also known as pixel values. In a natural image, there are certain structures (patterns) among these pixels, and it is such structures that constitute the appearance of the image.

In this work, a new image model is developed to characterize such structures. The development here focuses on the modeling of local structures, which we consider as a key aspect that characterizes images. Particularly, by integrating a probabilistic manifold to capture local pixel patterns and a Markov random field to bring them together into a joint formulation, this model is able to express rich local structures while maintaining the coherence between them.

We derive a variational EM algorithm to estimate the model from training images, as well as inference algorithms for inferring underlying images from partially corrupted observations. These inference algorithms are applied to solve various low-level vision problems, which particularly include image denoising and image inpainting. Experimental results will be presented on both tasks, with comparison to other MRF-based methods.

3.1 Probabilistic Image Models

Image modeling, which describes the structure and content of an image through a mathematical formalism, lies at the heart of computer vision and provides important basis for many vision tasks. In digital image processing and computer vision, a visual scene is typically represented as an image or a video, which, from a mathematical standpoint, can be considered as a function that maps each physical point (often called a *pixel*) on the image plane to an intensity or color value. In this thesis, we focus on intensity images, and thus each image can be represented as a real valued function as $I : \mathcal{D} \to \mathbb{R}$, where $\mathcal{D}$ is the image domain, *i.e.* the set of visible pixel coordinates. However, it would be straightforward to extend the work presented in this chapter to color images.

Generally, the pixel values of an image at different locations are strongly correlated with each other and often exhibit certain structures, which distinguish an image of natural scenes from arbitrary two-dimensional signals. Therefore, one key to image modeling is to capture such structures. With different motivations, researchers have developed various approaches to image modeling, among which many can be formulated as a parametric distribution over images, as $p(I;\theta)$, where θ denotes the parameters. This distribution is often called an *image prior*.

Given an image prior, many low-level vision tasks can be readily solved via Bayesian inference. Take image denoising for example. Suppose the observed image $\tilde{I}$ is generated from an underlying natural image I through a noisy measurement process as $p(\tilde{I}|I;\eta)$, where η denotes the parameter that characterizes the measurement (*e.g.* the variance of noise). Then, the task of recovering the underlying image can be formulated to be a Maximum-a-Posteriori optimization problem as below

$$\hat{I} = \operatorname*{argmax}_I p(I|\tilde{I};\theta,\eta) = \operatorname*{argmax}_I p(I;\theta)p(\tilde{I}|I;\eta). \tag{3.1}$$

Here, $\hat{I}$ denotes the optimal solution to this problem. Instead of pursuing the single optimum, one may also choose to characterize the inherent uncertainty, *e.g.* sampling from the posterior distribution $p(I|\tilde{I};\theta,\eta)$.

The task of image modeling and that of using an image model to accomplish vision tasks, as in Eq.(3.1), consists of three aspects:

1. Establish an image prior $p(I;\theta)$. This has been one of the central topics in computer vision for years. Consider the image recovery problem, prior knowledge about the structure or characteristics is generally needed to infer the missing or corrupted parts of an image. In previous work, two families of approaches are widely used to formulate image priors, namely the *manifold-based models* and the *MRF-based models*. As we shall see, they are complementary, and can potentially be integrated to derive a more effective model.

2. Formulate the measurement process, which various across different applications. In image denoising, it is a common practice to assume the observed images are corrupted by additive white noise, as

$$\tilde{I}(x) = I(x) + \varepsilon_x, \quad \varepsilon_x \sim \mathcal{N}(0, \sigma^2), \quad \forall x \in \mathcal{D}. \tag{3.2}$$

   Whereas this formulation may tend to oversimplify the actual measurement process, which, in many practical cases, produces results comparable with those produced using more sophisticated measurement models, with lower computational complexity.

3. Develop algorithms to solve the optimization problem in Eq.(3.1), or to sample from the posterior distribution. We will elaborate on this part later in this chapter, when the image model is established.

The primary goal of this chapter is to develop a new image prior to model natural images. Before introducing the new model, we first briefly review previous approaches to this problem, which, as mentioned, roughly fall into two categories: *manifold-based* and *MRF-based*. As we shall see later, these two types of approaches have their respective strength and weakness, and our new approach is motivated by combining the strengths of both.

3.1.1 Manifold-based Image Modeling

An important family of approaches to image modeling is based on manifolds. Specifically, an image with N pixels can be treated as a vector in $\mathbb{R}^N$. Here, to simplify the discussion, we suppose all images to be modeled are of the same size, and thus their vector representations are in the same space.

It has long been observed that natural images mostly concentrate around a low-dimensional manifold. Motivated by this, different methods have been developed to estimate such a manifold from training images. Among these methods, there is an important family – subspace modeling, which is based on a further simplified assumption, that is, images lie on a low-dimensional subspace. In a subspace model, an image can be expressed as a linear combination of base images, as

$$I = \mathbf{B}\mathbf{c} + \boldsymbol{\varepsilon}. \tag{3.3}$$

Suppose the dimension of the subspace is q, then $\mathbf{B}$ is a matrix of size $N \times q$, of which each column corresponds to a base image. $\mathbf{c} \in \mathbb{R}^q$ is a coefficient vector.

Many methods have been proposed to learn $\mathbf{B}$, the image bases, from a set of training images. Representative methods include Principal Component Analysis (PCA) [104, 105], Independent Component Analysis (ICA) [10], as well as others that rely on information theoretical criteria.

The linear assumption underlying the subspace models tends to be too strong for many practical problems, even for those only involving a restricted class of images. In later work, more sophisticated methods that allow the modeling of nonlinear manifolds are developed. These methods extend the classic linear models in different ways:

1. *Locally linear approximation.* An important way to construct a manifold is through locally linear approximation, that is, to use a set of linear spaces, each covering a small region on the manifold. Different methods use different ways to characterize the locally linear spaces. A well-known method in this family is the one proposed by Roweis and Saul, called Locally linear embedding (LLE) [86].

This algorithm maps each image to a low dimensional point, such that each point can be approximate by a linear combination of its neighbors, using the same coefficients as those for the high dimensional representation. In this way, the manifold embedding found in this way is expected to preserve the local relations between samples. Another well known method called Locality preserving projection (LPP) [45] learns the optimal projection onto the embedded manifold by finding the best approximations to the eigenfunctions of its Laplacian Beltrami operator.

2. *Nonlinear mapping via kernels.* Many subspace-based models can be readily generalized to describe nonlinear manifold through the "kernel trick". The basic idea is to learn a subspace of a new representation space obtained through a nonlinear mapping induced by a positive definite kernel. This new representation space is called the *Reproducing Kernel Hilbert Space*, which may have infinite dimension. However, in most cases, this space does not need to be explicitly instantiated. All computations can still be performed in the original space, where the evaluation of inner products are replaced by the evaluation of kernels.

   Using this technique, linear subspace models can be extended to nonlinear ones (*e.g.* Kernel PCA). This strategy has been widely used in vision tasks, such as face recognition [117] and object detection [3].

3. *Generalized PCA.* Vidal *et al.* proposed the Generalized Principal Component Analysis (GPCA) [107], which is an algebraic geometric extension to the PCA method. This algorithm aims to find an indefinite number of subspaces of varying dimensions from sample data points. Here, the subspaces are represented by a set of homogeneous polynomials, such that the degree of these polynomials is equal to the number of subspaces, and the gradients are orthogonal to the given subspaces at each point.

While these methods are effective for describing the appearance of objects in specific classes (*e.g.* faces), they have several limitations, making them unsuitable for

generic image modeling

1. The models constructed based on manifolds typically provide a holistic characterization of the images to be modeled. With a low-dimensional internal representation, they can only express certain structural variations. However, the structure of a generic image, especially those containing multiple objects, is subject to substantial variations. While some modeling extensions (*e.g.* mixture models) may help to extend the model's capability of expressing such variations, it is still far beyond any current model's capability to give a holistic characterization of generic images, except in a very restricted context.

2. Generic images are typically in a very high dimensional space, over which, learning subspace or manifold parameters often requires a huge amount of training samples to achieve a reasonable reliability. For image modeling, the problem is even more challenging, as patterns of very different structures can be presented in an image. Therefore, directly learning a manifold over the holistic appearance of generic images would be extremely difficult, where an exceedingly large number of samples may be needed to reliably estimate a model.

3.1.2 MRF-based Image Modeling

Markov random fields (MRFs) provide a generic probabilistic formulation for low-level image modeling. Unlike manifold-based models, MRFs emphasize local coherence rather than global structure. Generally, an MRF model defines the probability density function through a set of potential functions, each over a clique, as follows.

$$p(I;\theta) = \frac{1}{Z}\prod_{i=1}^{n}\phi_i(I(c_i);\theta_i). \tag{3.4}$$

Here, c_i is the i-th clique, which may contain a single pixel or cover several neighboring pixels. Associated with this clique is a potential function ϕ_i, which characterizes the statistical relations between the pixels of the clique. Z is a normalization constant, whose value depends on the parameter θ. Section 2.1 provides a more detailed review

of MRFs.

Whereas MRF-based image models all share the same form as given by Eq.(3.4), they may have different potential functions. A classic formulation of MRF that has been widely used in low-level vision employs pairwise quadratic potentials, as

$$p(I;\theta) = \frac{1}{Z}\exp\left(-\frac{\theta}{2}\sum_{(i,j)\in \mathrm{Nbs}}(I(x_i) - I(x_j))^2\right). \tag{3.5}$$

This model is motivated by a simple observation that neighboring pixels tend to have similar values. This formulation actually defines a joint Gaussian distribution over the pixel values, and thus is a *Gaussian MRF*.

Simplicity is probably one of the most important advantage of a Gaussian MRF. Let $\mathbf{L}$ be the Laplacian matrix of the neighbor graph (*i.e.* the graph with edges between neighboring pixels), then Eq.(3.5) can be rewritten in a matrix form as

$$p(I;\theta) = \frac{1}{Z}\exp\left(-\frac{\theta}{2}\mathrm{vec}(I)^T\mathbf{L}\mathrm{vec}(I)\right). \tag{3.6}$$

Here, $\mathrm{vec}(I)$ denotes the vector obtained by stacking all pixel values of I. For an image denoising task, if Gaussian white noise is assumed for the measurement process, as in Eq.(3.2), then there is an analytic solution to the MAP problem given by Eq.(3.1), as below

$$\mathrm{vec}(\hat{I}) = (\theta\mathbf{L} + \sigma^{-2}\mathbf{I})^{-1}(\sigma^{-2}\mathrm{vec}(\tilde{I})). \tag{3.7}$$

However, in natural images, the distribution over high frequency components often exhibits heavy tailed characteristics, which Gaussian MRFs are not able to capture. Consequently, methods relying on Gaussian MRFs tend to blur edges and contours when applied to image recovery tasks.

This problem has been widely known in computer vision community. To better preserve sharp discontinuities in images, people have proposed potential functions in other forms that have heavier tails [36]. Though partly alleviating the issue of over blurring, such formulations are still very limited in their expressiveness, as they only consider pairwise relations and do not explicitly take higher order interactions into

account.

Following Zhu *et al.*'s pioneering work [119], a series of high order MRFs has been proposed for image modeling. [119] presents a model called FRAME, which combines filtering theory and MRF modeling to characterize images of homogeneous texture patterns. In this model, A set of filters is selected from a general filter bank and applied to training images, and the histograms of the filtered images are extracted. These histograms are estimates of the marginal feature distributions. Then, the maximum entropy principle is employed to construct the joint distribution $p(I)$ over texture images, which is restricted to have the same marginal feature distributions. Zhu *et al.* [119] showed that this joint distribution can be expressed in form of a Markov random field, as

$$p(I) = \frac{1}{Z} \exp\left(-\sum_{k=1}^{K} \lambda_k \sum_{c \in \mathcal{C}} f_k^T I(c) \right). \tag{3.8}$$

Here, K is the number of selected filters, and $f_1, \ldots, f_K$ are the filter kernels. $\mathcal{C}$ is the set of all cliques, each covers the support of a filter kernel. Hence, $f_k^T I(c)$ can be understood as the response of filter f_k at patch c.

Later, Roth and Black [83, 85] proposed the Field of Experts (FoEs), which extends the FRAME model by formulating local potentials as products of experts. Specifically, the joint probability density function is given by

$$p(I) = \frac{1}{Z} \exp\left(\sum_{k=1}^{K} \sum_{c \in \mathcal{C}} \phi_k(f_k^T I(c); \alpha_k) \right). \tag{3.9}$$

Here, the filters $f_1, \ldots, f_K$ are learned rather than being selected from a pre-defined filter bank. Also, the potential function ϕ_k is designed to capture heavy-tailed characteristics. In [85], a differentiable approximation of L1-norm as below is used

$$\phi(u; \alpha, \beta) = \alpha \sqrt{\beta + u^2}. \tag{3.10}$$

In [84], Roth and Black further developed a Steerable Random Fields, where steerable

filters are used and potentials are defined upon steered filter responses. Current methods that utilize MRFs for natural image modeling are limited in two aspects.

1. First, many methods rely on the distributions of filter responses to derive clique potentials, obscuring some aspects of the generative model. As we shall see in next section, such models have limited capacity to describe local patterns.

2. Second, non-Gaussian potentials, which are often used to capture heavy-tailed characteristics, usually lead to computational difficulties in both learning and inference. For example, *Contrastive divergence sampling*, which is known to converge very slowly, is used for maximum likelihood estimation of the Field of Experts model in [83].

   Consequently, a variety of approximate formulations have been proposed. Weiss and Freeman [115] derived tractable lower and upper bounds of the partition function of the Field of Experts model when Gaussian potentials are used, such that more efficient optimization-based methods can be employed to solve the problem. They also extended the results to non-Gaussian potentials and developed an approximate method to obtain the maximum likelihood estimation. Tappen [33] adopts a strategy called variational mode learning, where rather than maximizing the likelihood of the training data, the MRF parameters are found by minimizing a loss function that measures the difference between the ground-truth image and optimal image under the MRF model. Samuel and Tappen [87] proposed a variant of the Field of Experts, where the MRF model is trained by optimizing the parameters so that the minimum energy solution of the model is as similar as possible to the ground-truth.

Recent work [34, 98] suggests the use of conditional random fields (CRFs) that directly model the posterior instead of the prior, which substantially improves the learning efficiency under certain settings. However, as articulated by Schmidt *et al.* [88], the gain in efficiency often comes with the loss of generality.

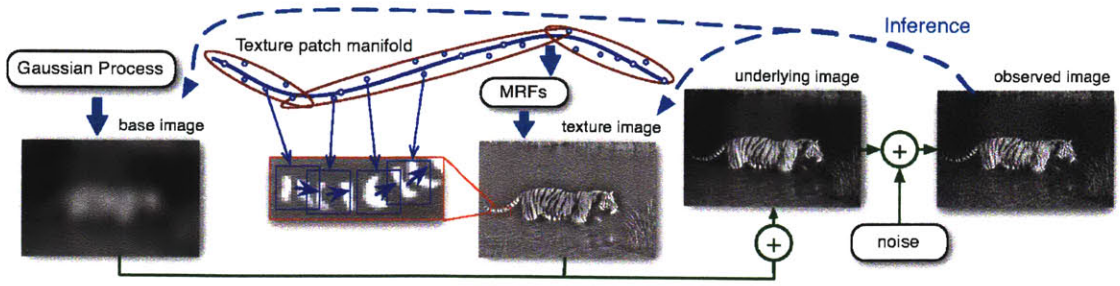


Figure 3-1: The overall framework of the generative image model. Here, each image is considered as a combination of a base image that roughly reflects the smooth lighting variation, and a texture image that captures the local details. The base image is generated from a prior formulated in the form of a Gaussian process; while the texture image is generated as a composite of oriented local patches drawn from the patch manifold. A Markov random field conditioned on the local patches is introduced to produce the entire image, which explicitly enforces coherence across patches. This figure also illustrates how this model can be applied to image denoising. Specifically, given a learned model, the variational inference algorithm will incorporate both the prior knowledge provided by this model and the observed noisy image to derive the posterior distribution over the MRFs, and thus recover the underlying image in a Bayesian fashion.

3.2 A New Image Prior

In this work, we develop a new image prior, motivated by the following observation. The global appearance structure of natural images varies dramatically from image to image and from scene to scene. One key aspect shared by natural images that distinguish them from other two-dimensional signals is the structures of the local patterns.

The new image model is a probabilistic generative model, which comprises a patch model that leverages the expressive power of manifold modeling to capture the variations of local patterns, and a family of Markov random fields to enforce coherence across patches. Specifically, to produce an image, local components from the patch manifold are selected to generate individual patches, and thereon a conditional MRF is constructed to generate an image coherently.

Figure 3-1 shows the overall framework of the proposed image prior. Here, an image I is considered as the superposition of two components: (1) a *low-frequency component* B, called the *base image*, which roughly reflects smooth lighting variation

over the entire image (*e.g.* which region is dark and which is lighter); and (2) a *high-frequency component* Y, called the *texture image*, which captures the local details and is modeled as a coherent composite of local texture patterns. Through such a decomposition, the effect of overall illumination variation can be roughly separated from the modeling of local patterns (*e.g.* textures).

3.2.1 Modeling Base Images

Intuitively, a *base image* is simply an excessively blurred version of the original image. We formulate the prior distribution of the base images as a Gaussian process, which, with a proper choice of covariance function, is effective in modeling smooth signals. Particularly, the covariance function that we use here is defined to be

$$\mathrm{Cov}(B(x), B(x')) = a_B \exp\left(-\frac{\|x - x'\|^2}{2\sigma_B^2}\right). \tag{3.11}$$

Here, x and x' are the coordinates of two pixels, and $\mathrm{Cov}(B(x), B(x'))$ is the prior covariance between the corresponding pixel values. The parameter a_B and σ_B^2 can be learned from training images. In particular, σ_B^2 controls the range of correlation. Generally, a model with larger value of σ_B^2 would enforce longer range of coupling, thus generating more blurry images.

While more sophisticated models might be used to describe the base image, we did not choose to pursue this direction further. The reason is that for many low-level vision tasks that this work is targeting, the key is to recover the local structures, and thus this simple Gaussian process model is sufficient. That being said, it would be interesting, as a future work, to study the modeling of global image structures (*e.g.* spatial configurations of regions) and see how it might contribute to vision applications.

3.2.2 The Patch Manifold Model

The generative model of texture images is comprised of two main components: (1) *a probabilistic patch manifold model* that aims to capture the structures of local

patterns, and (2) *a conditional MRF* that enforces coherence across patches.

First of all, a generative patch model is introduced, which characterizes local patterns at the level of patches. The construction of this model is motivated by two observations:

1. In natural images, intensity values of neighboring pixels are highly correlated. Let d_p be the dimension of a patch vector (*i.e.* the number of pixels in a patch), then most patches may lie around a manifold of dimension much lower than d_p. Hence, the task of modeling the distribution of patches can be partly reduced to the estimation of such a patch manifold.

2. A patch and its rotated versions are equally likely for a natural image. This might not be necessarily true in practice. However, based on this assumption, we may substantially reduce the complexity of the manifold by mapping all rotated versions to a single point on the manifold.

The generation of a patch based on this model consists of three steps:

1. Generate a *canonical patch* from a component of the manifold. Here, a *canonical patch* is a patch with standard orientation.

2. Generate a rotated version of the canonical patch.

3. Generate the residues. This step allows deviation from the manifold.

Next, I will discuss these steps in detail.

Generation of canonical patches

Patches that are rotated versions of each other are considered to be equivalent. Given an equivalence class of patches, we designate the patch with horizontal orientation as the *canonical patch* of this class. Here, the orientation of a patch is determined by the leading eigenvector of the *structure tensor* [11]. In particular, the structure

tensor associated with a patch p is defined to be

$$\mathbf{S}(p) = \begin{bmatrix} \sum_{x \in p} g_h^2(x) & \sum_{x \in p} g_h(x) g_v(x) \\ \sum_{x \in p} g_h(x) g_v(x) & \sum_{x \in p} g_v^2(x) \end{bmatrix}, \tag{3.12}$$

where $g_h(x)$ and $g_v(x)$ are respectively the horizontal and vertical component of the image gradient at pixel x. Generally, one may use other methods to determine the principal orientation. The reason that we use structure tensor here is that it is robust again noise and is easy to implement.

Canonical patches are described by a manifold of dimension $d_m < d_p$. As a patch may exhibit very different patterns, this manifold is nonlinear, which we approximate using a mixture of locally linear components. Each component here covers a subset of similar patterns. Specially, these components are formulated as d_m-dimensional hyperplanes, denoted by $H_1, \ldots, H_K$. Each hyperplane $H_k = (\boldsymbol{\mu}_k, \mathbf{W}_k)$ is characterized by an offset vector $\boldsymbol{\mu}_k \in \mathbb{R}^{d_p}$ and a basis matrix $\mathbf{W}_k \in \mathbb{R}^{d_p \times d_m}$. With these notations, each canonical patch $\mathbf{x}$ in on the hyperplane H_k can be expressed as

$$\mathbf{x} = \boldsymbol{\mu}_k + \mathbf{W}_k \mathbf{z}. \tag{3.13}$$

This mixture model has a prior categorical distribution $\boldsymbol{\pi}$ over the constituent hyperplanes. To generate a canonical patch, one can first choose a specific hyperplane H_k from $\boldsymbol{\pi}$, then draw the *latent representation* $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, and finally obtain a patch as in Eq.(3.13). Here, the dimension of $\mathbf{z}$ is d_m.

Patch rotation

The patch that we actually observe in an image is a rotated version of the canonical patch generated from the manifold. To generate this rotated version, one first draws an orientation ω from a uniform distribution over $[0, 2\pi]$, and then rotates the canonical patch in the clockwise direction. The resultant patch is denoted by $R(\mathbf{x}, \omega)$.

While we limit this analysis to relations, other geometric transforms can be incorporated. By considering larger equivalence classes, the complexity of the canonical

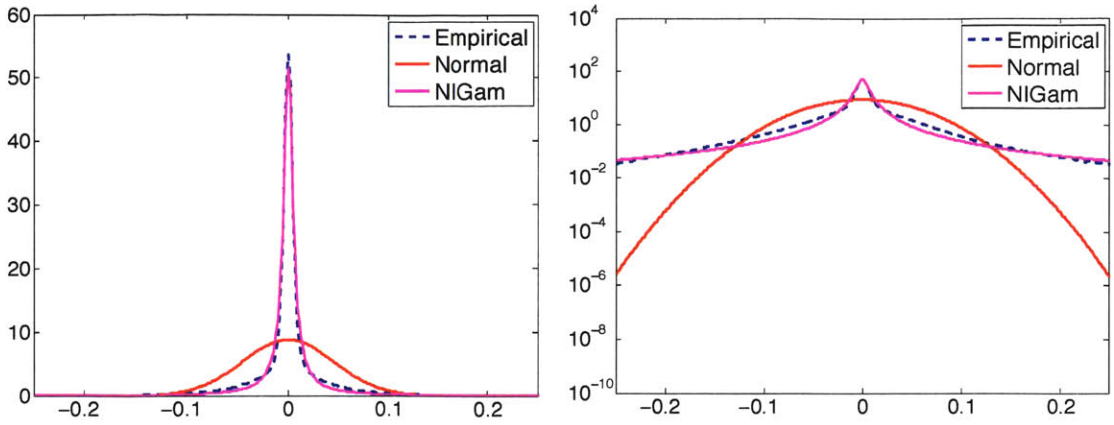


Figure 3-2: This figure compares how well normal distribution and normal inverse-gamma distribution fit the pixel-wise residues. The left and right figures respectively show the estimated models against the empirical distribution in linear and log-scale.

patch manifold may be further reduced. However, one should be cautious when using other transforms. For example, the use of size scaling may lead to difficulties when the size of a patch is fixed.

Generation of residues

The model allows small deviation from the manifold via a residue term. In order to select a suitable residue distribution, we fit a mixture of hyperplanes to a set of patches extracted from natural images and examine the marginal distribution of pixel-wise residues. Empirical analysis reveals heavy-tailed characteristics.

A variety of models can be used to approximate a distribution with heavy-tailed characteristics. A model that has been widely used is the *Gaussian Scale Mixture (GSM)*. In general, its probability density function is defined to be

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; 0, \sigma_k^2) = \sum_{k=1}^{K} \frac{\pi_k}{\sqrt{2\pi\sigma_k^2}} e^{-x^2/(2\sigma_k^2)}. \tag{3.14}$$

To use this formulation, one has to first specify the value of K, the number of components. Also, this model has $2K$ parameters to estimate, including the prior weights and the variances of components.

We find that the *normal inverse-gamma distribution*, a simpler model, performs

equally well in practice. A normal inverse-gamma distribution can be viewed as a *continuous Gaussian scale mixture*, where the variances are generated from an inverse-gamma distribution. Specifically, this distribution is controlled by only two parameters: a shape parameter α_r and a scale parameter β_r, and it is denoted by $\text{NIGam}(\alpha_r, \beta_r)$. Sampling $\xi \sim \text{NIGam}(\alpha_r, \beta_r)$ is as follows:

$$\sigma_r^2 \sim Inv.Gamma(\alpha_r, \beta_r), \quad \xi \sim \mathcal{N}(0, \sigma_r^2). \tag{3.15}$$

The probability density function of this distribution is given by

$$p_{\text{NIGam}}(\xi; \alpha_r, \beta_r) = \frac{1}{2\pi} \frac{\Gamma(\alpha_r + 1/2)}{\Gamma(\alpha_r)} \beta_r^{\alpha_r} \left(\beta + \frac{\xi^2}{2}\right)^{-(\alpha_r + 1/2)}. \tag{3.16}$$

We can see that the pdf value attenuates as a power function with a fixed exponent $-(\alpha_r + 1/2)$ as ξ increases. Clearly, a normal inverse-gamma distribution has as a heavier tail than a normal distribution. When $\alpha_r > 1$ and $\beta_r > 0$, the variance of ξ is given by $\beta_r/(\alpha_r - 1)$.

Figure 3-2 shows that the normal inverse-gamma distribution yields much better fit to the empirical distribution of the pixel-wise residual values. Furthermore, as we shall see in the next section, the conjugacy between inverse-gamma and normal distribution (*w.r.t.* the variance) leads to close-form updates in the variational inference procedure.

The Overall Formulation

Altogether, we obtain a graphical model to generate patches, as illustrated in Figure 3-3. Here is a brief summary of the model. This model comprises K hyperplanes $H_1, \ldots, H_K$ to approximate the patch manifold. Each hyperplane H_k is characterized by a basis matrix $\mathbf{W}_k$ and an offset vector $\boldsymbol{\mu}_k$. In addition, there is a discrete distribution $\boldsymbol{\pi}$ over these components, and a normal inverse-gamma distribution to model the residues, with parameters α_r and β_r.

For each local clique c of an image, a patch is generated from this manifold, through the process summarized below.

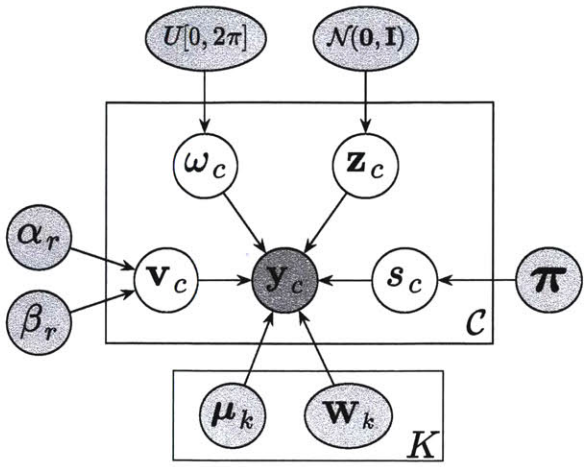


Figure 3-3: This is the graphical model for generating patches. In this model, the generation of a patch $\mathbf{y}_c$ consists of four steps: (1) choose a component $s_c \sim \boldsymbol{\pi}$; (2) generate the latent representation $\mathbf{z}_c \sim \mathcal{N}(0, \mathbf{I})$, and thus the canonical patch $\mathbf{x}_c = \mathbf{W}_{s_c}\mathbf{z}_c + \boldsymbol{\mu}_{s_c}$, (3) draw an orientation ω_c and rotate the patch accordingly, obtaining $R(\mathbf{x}_c, \omega_c)$, (4) generate the residue vector $\boldsymbol{\xi}_c$, by drawing each entry independently from $\text{NIGam}(\alpha_r, \beta_r)$, and add the residues to the patch.

1. Choose a particular component of the manifold, by drawing its indicator $s_c \sim \boldsymbol{\pi}$.

2. Generate the latent representation $\mathbf{z}_c \sim \mathcal{N}(0, \mathbf{I})$. Then, the canonical patch is given by $\mathbf{x}_c = \mathbf{W}_{s_c}\mathbf{z}_c + \boldsymbol{\mu}_{s_c}$.

3. Draw an orientation $\omega_c \sim U([0, 2\pi])$, and generate the rotated version $R(\mathbf{x}_c, \omega_c)$.

4. Generate the residue $\boldsymbol{\xi}_c = (\xi_c^{(1)}, \ldots, \xi_c^{(d_p)})$. Each entry $\xi_c^{(i)}$ here is independently sampled from the normal inverse gamma distribution, as $\xi_c^{(i)} \sim \text{NIGam}(\alpha_r, \beta_r)$.

With all these variables, we can obtain a patch as

$$\mathbf{y}_c = R(\mathbf{x}_c, \omega_c) + \boldsymbol{\xi}_c = R(\mathbf{W}_{s_c}\mathbf{z}_c + \boldsymbol{\mu}_{s_c}, \omega_c) + \boldsymbol{\xi}_c. \tag{3.17}$$

As an empirical comparison, we collect $100,000$ patches of size 13×13, and estimate both a probabilistic manifold and a Field of Experts model over this set. Figure 3-4 shows the samples respectively generated from both models. Qualitatively, the patch manifold model developed here yields more structured patterns.

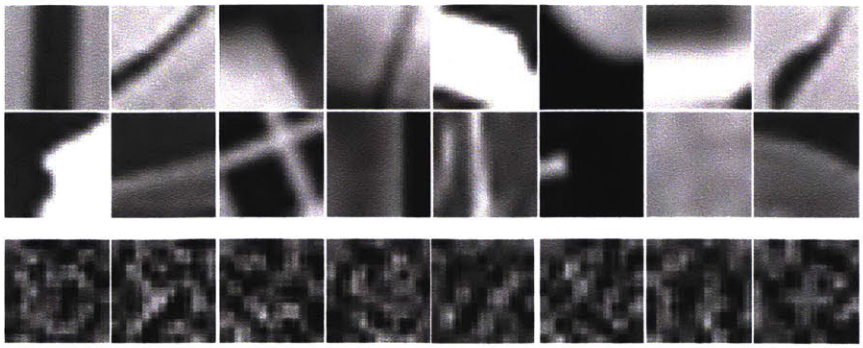


Figure 3-4: The first two rows show the sample patches drawn from the probabilistic patch manifold (the size of each patch here is 13×13). The last row shows the sample patch generated from the Field of Experts model [83] with 5×5 filter banks, which we obtained using a Gibbs sampler that runs on a 13×13 grid.

3.2.3 Patch Coherence via Markov Random Fields

A critical element of the proposed model is to maintain coherent image structure across overlapping image patches. A simple idea to improve coherence across patches is *blending*, that is, to generate overlapping patches independently and combine them with smoothly varying weights in regions where patches overlap. Simple methods as such may yield noticeable artifacts when there is inconsistency between neighboring patches. Alternately, image quilting [29] addresses this issue by finding the optimal boundary between patches via minimum error boundary cut. However, this requires solving a discrete optimization for all overlapping patches and is not easily incorporated into a probabilistic generative model.

The proposed framework uses a conditional MRF to enforce coherence across patches. Consider an image Y with a collection of overlapping patches, denoted by $\mathcal{C}$. For each patch $c \in \mathcal{C}$, we denote the vector of pixel values in c by $\mathbf{y}_c$. Note that $\mathbf{y}_c$ and $\mathbf{y}_{c'}$ may share part of the values when c and c' overlap.

Given the patch model, we generate an image through the following procedure with two stages.

Stage 1: Generation of latent variables

Recall that generating a patch $\mathbf{y}_c$ from the probabilistic patch manifold model involves several latent variables:

1. an indicator $s_c \sim \boldsymbol{\pi}$ that specifies a component H_{s_c} to generate the patch;
2. a latent (low-dimensional) representation $\mathbf{z}_c \in \mathbb{R}^q$;
3. an orientation $\omega_c \in U([0, 2\pi])$ for patch rotation;
4. a vector of variances $\mathbf{v}_c = (v_c^{(1)}, \ldots, v_c^{(d_p)}$ used to generate the pixel-wise residues.

These variables together as $(s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c)$ are called the *local configuration* for patch $\mathbf{y}_c$. In this model, they are generated independently for each patch.

Stage 2: Generation of the texture image Y from an MRF

Instead of generating each patch $\mathbf{y}_c$ independently based on the local configuration, we construct an MRF over the entire image conditioned on local configurations for all patches and sample and generate an image therefrom. The formulation of this MRF is given by

$$p(Y|\mathbf{s}, \mathbf{z}, \boldsymbol{\omega}, \mathbf{v}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi(\mathbf{y}_c|s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c). \tag{3.18}$$

Here, the potential value $\phi(\mathbf{y}_c|\mathbf{s}_c, \mathbf{z}_c, \boldsymbol{\omega}_c, \mathbf{v}_c)$ is defined to be the conditional pdf of $\mathbf{y}_c$ *w.r.t.* the patch model introduced above, and Z is a normalization constant. In particular, the potential function ϕ is given by

$$\phi(\mathbf{y}_c|\mathbf{s}_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c) = \prod_{j=1}^{d_m} \frac{1}{\sqrt{2\pi(v_c^j)^2}} \exp\left(-\frac{(R(\mathbf{x}_c, \omega_c)^{(j)} - \mathbf{y}_c^{(j)})^2}{2(v_c^j)^2}\right). \tag{3.19}$$

Here, $\mathbf{x}_c = \mathbf{W}_{s_c}\mathbf{z}_c + \boldsymbol{\mu}_{s_c}$, and $v_c^j \sim Inv.Gamma(\alpha_r, \beta_r)$. Also, a superscript (j) is used to indicate the j-th pixel of a patch. For the convenience of computation, we reformulate the potential. Instead of rotating the canonical patch generated from the manifold, we rotate the observed patch in reverse direction and compare it with the canonical patch. Though both are equivalent, the latter simplifies inference, as $\mathbf{x}_c$

involves a latent representation $\mathbf{z}_c$ that needs to be inferred. The resultant potential is thus given by

$$\phi(\mathbf{y}_c|\mathbf{s}_c, \mathbf{z}_c, \boldsymbol{\omega}_c) = \prod_{j=1}^{d_m} \frac{1}{\sqrt{2\pi(v_c^j)^2}} \exp\left(-\frac{(R(\mathbf{y}_c, -\omega_c)^{(j)} - \mathbf{x}_c^{(j)})^2}{2(v_c^j)^2}\right). \tag{3.20}$$

It is important to note:

1. This MRF, including the parameters of the potential functions and the value of the normalization constant, depends on local configurations. That's the reason we call it a *conditional MRF.*

2. With this MRF formulation, the texture image Y is generated as a whole by sampling from the MRF model, which is different from sampling individual patches and combining them through a post-processing procedure. Intuitively, one may see this as a process that couples the generation of all patches.

Substituting this potential function given by Eq.(3.20) into Eq.(3.18), we can rewrite the likelihood of Y conditioned on local configurations as

$$p(Y|\mathbf{s}, \mathbf{z}, \boldsymbol{\omega}, \mathbf{v}) \propto \exp\left(-\sum_{c\in\mathcal{C}} E_c(\mathbf{y}_c|s_c, z_c, \omega_c)\right). \tag{3.21}$$

Here, the energy term associated with patch c is given by

$$\begin{aligned} E_c &= \frac{1}{2}\sum_{j=1}^{d_m}(v_c^j)^{-1}\left(R(\mathbf{y}_c, -\omega_c)^{(j)} - \mathbf{x}_c^{(j)}\right)^2 \\ &= \frac{1}{2}\sum_{j=1}^{d_m}(v_c^j)^{-1}\left(R(\mathbf{y}_c, -\omega_c)^{(j)} - (\mathbf{W}_{s_c}\mathbf{z}_c + \boldsymbol{\mu}_{s_c})^{(j)}\right)^2. \end{aligned} \tag{3.22}$$

As the energy term is quadratic, the MRF constructed above is a Gaussian MRF. It is worth emphasizing again that this MRF is conditioned on local configurations. Integrating out the variances v_c^j, we will end up with a continuous mixture of MRFs with heavy-tailed marginals on the residues.

Figure 3-5: This figure, depicting three overlapping patches (green, red, and green from left to right), illustrates how inter-patch coherence is ensured. On the left is a small part of a natural image. By flipping the rightmost patch, we obtain the image on the right. Whereas the rightmost patch may be captured by the manifold, the innermost patch (red) has a discontinuity and as such is unlikely to be well explained by the manifold. Hence, by driving all patches towards the manifold, the MRF favors coherence across the left, middle and right patches.

Based on the MRF derived above, the maximum-a-posterior inference will drive each patch towards the patch manifold. As patches overlap with each other, if two adjacent patches are inconsistent, the patch overlap with both would be unlikely to be generated from the patch manifold. Hence, this process, via patch overlapping, also encourages coherence across patches (see Figure 3-5).

3.2.4 The Joint Likelihood

Overall, the model has the following parameters: (1) the hyperplanes of the manifold: $H_1, \ldots, H_K$ with $H_k = (\boldsymbol{\mu}_k, \mathbf{W}_k)$, (2) the prior $\boldsymbol{\pi}$ over these hyperplanes, and (3) the parameters of the residue distribution α_r and β_r. These parameters together are denoted by $\boldsymbol{\theta}$. In addition, each texture image Y is associated with several hidden variables: the hyperplane selectors $\mathbf{s}$, the latent representations $\mathbf{z}$, the orientations $\boldsymbol{\omega}$, and the residue variances $\mathbf{v}$. Given $\boldsymbol{\theta}$, the joint likelihood of Y and these hidden variables is

$$p(Y|G_Y) \prod_{c \in \mathcal{C}} p(s_c|\boldsymbol{\pi}) p(\mathbf{z}_c) p(\omega_c) p(\mathbf{v}_c|\alpha_r, \beta_r). \tag{3.23}$$

Here, $p(s_c|\boldsymbol{\pi})$ is a categorical distribution, $p(\mathbf{z}_c)$ is a standard Gaussian distribution, $p(\omega_c)$ is a uniform distribution over $[0, 2\pi]$, and $p(\mathbf{v}_c|\alpha_r, \beta_r)$ is a multivariate inverse-gamma distribution. G_Y denotes the conditional MRF to generate Y, given

by Eq.(3.21) and Eq.(3.22).

Discussions

We discuss some issues with respect to the image prior presented above.

1. This model focuses on local characteristics. This is sufficient for low level vision tasks where recovery of local patterns is the main objective. Consider an image corrupted by white noise, its overall appearance structure is largely intact. Denoising such an image mainly requires prior knowledges on local textures.

2. In the generative model described above, we actually establish a prior over a space of Gaussian MRFs, in which each MRF is conditioned on a configuration of local patch models. This contrasts with previous work utilizing a single MRF or CRF (either hand-crafted or learned) for low-level vision tasks [83, 84, 115].

   Formulating the image prior as a distribution over MRFs brings forth several benefits: (1) a probabilistically consistent generative model; (2) the capacity to model heavy tailed characteristics or other statistical properties that are not well described by Gaussian models; and (3) the availability of efficient algorithms for learning and inference.

3. Though assumed independent a priori, the local configurations of different patches will be coupled given the observations[1], provided that the patches are overlapping. The inference procedure will take the information from the observed image to guide the choices of latent values, encouraging the generation of locally coherent images that have similar appearance structure as the observation.

   That being said, we believe that there is interesting dependence among these local configurations, which is worth further investigation as future work.

[1]Section 2.1 discusses conditional independence of graphical models, which contains an important result for Bayesian networks: parent nodes of an observed node are mutually dependent in the posterior distribution.

4. The local model of each patch is similar to a mixture of PPCA that has been employed for digit recognition and image compression [101]. The novelty here consists in the maintenance of coherence across patches via conditional MRFs, and the use of dominant orientations and heavy-tailed residue distribution.

5. Using manifold model to derive clique potentials distinguishes it from previous work on natural image modeling, where the use of derivative filters in defining potentials is a common practice.

3.3 Learning the Image Model

In practice, we can learn the model parameters from a given set of training images. Before describing the details of the learning algorithm, we first introduce our experiment settings. Specifically, our experiments are performed on the Berkeley Segmentation Data Set and Benchmarks 500 (BSDS500) [5], which has been widely used to assess denoising and inpainting methods[83, 84, 115]. Note that we use BSDS500[5], a recently released extension including 200 new test images.

This data set specifies a subset of 200 images for model training, which we denote by $I_1, \ldots, I_n$. Here, $n = 200$. The algorithm first decomposes each image I_i into two components: a base image B_i and a texture image Y_i via simple image processing. In particular, for an image I_i, a low-pass filter[2] is used to produce an excessively smoothed version, which is treated as the base image B_i, then the texture image is set to be $Y_i = I_i - B_i$. In this way, a set of base images $B_1, \ldots, B_n$ and a set of texture images $Y_1, \ldots, Y_n$ are derived, which are then respectively used to learn the Gaussian process prior and the patch manifold model. Below, we will respectively describe the detailed procedures.

[2]A filter with a Gaussian kernel of large radius ($\sigma = 25$ pixels) is used in our experiments to obtain the base image.

3.3.1 Learning the Gaussian Process Prior

The estimation of the GP prior is based on all base images derived as above. These base images are assumed to be independently generated from a Gaussian process, as

$$B_i \sim \mathrm{GP}(\mu_B, \kappa_B), \quad i = 1, \ldots, n. \tag{3.24}$$

Here, μ_B is a constant mean value, and κ_B is a covariance function defined by

$$\kappa_B(x, y) = a_B \cdot \exp\left(-\frac{\|x - y\|^2}{2\sigma_B^2}\right). \tag{3.25}$$

Then, for an image I_i with h rows and w columns, this covariance function gives rise to a covariance matrix $\mathbf{C}_i$ of size $(hw) \times (hw)$, as

$$\mathbf{C}(u, v) = \kappa_B(x_u, x_v) = a_B \cdot \exp\left(-\frac{\|x_u - x_v\|^2}{2\sigma_B^2}\right). \tag{3.26}$$

Here, x_u and x_v are the coordinates of the u-th and v-th pixels. Let $\mathbf{b}_i = \mathrm{vec}(B_i)$ be the vector comprised of all pixel values in B_i. Then under this model, the probability density at B_i is given by

$$p(B_i; \mu_B, \kappa_B) = \frac{1}{\sqrt{(2\pi)^{hw/2}|\mathbf{C}_i|^{1/2}}} \exp\left(-\frac{1}{2}(\mathbf{b}_i - \mu_B)^T \mathbf{C}_i^{-1}(\mathbf{b}_i - \mu_B)\right). \tag{3.27}$$

Let N be the number of pixels in an image. The complexity of evaluating this pdf is $O(N^3)$, which is prohibitive for a typical image, where N is often over 10^5.

Here, we only have to estimate three parameters μ_B, a_B, and σ_B. Hence, it is not necessary to use the entire image to obtain reliable estimates. In our experiment, a simplified method is used instead. We first estimate μ_B, for which, the maximum likelihood estimate is simply the mean of all pixel values in all base images.

The estimation of a_B and σ_B can be found using numerical gradient ascent. To make the computation tractable, rather than using the whole image, we randomly draw a set of pixel pairs from the base images. Each pair is denoted by $((u_i, x_i), (v_i, y_i))$. Here, u_i and v_i are the pixel values minus μ_B, and x_i and y_i are

the pixel coordinates. Note that the two pixels in each pair must be from the same image. The marginal distribution of (u_i, v_i) remains a Gaussian distribution, as

$$p((u_i, v_i); \kappa_B) = \frac{1}{\sqrt{(2\pi)^{hw/2}|\mathbf{\Sigma}_i|^{1/2}}} \exp\left(-\frac{1}{2}((u_i, v_i))^T \mathbf{\Sigma}_i^{-1}((u_i, v_i))\right). \tag{3.28}$$

Here, $\mathbf{\Sigma}_i$ is the marginal covariance of (u_i, v_i), which is given by

$$\mathbf{\Sigma}_i = \begin{bmatrix} a_B & a_B e^{-d_i^2/(2\sigma_B^2)} \\ a_B e^{-d_i^2/(2\sigma_B^2)} & a_B \end{bmatrix}, \qquad \text{with } d_i = \|x_i - y_i\|. \tag{3.29}$$

Hence, we have

$$|\mathbf{\Sigma}_i| = a_B^2(1 - \rho_i(\sigma_B)^2), \qquad \text{with } \rho_i(\sigma_B) = \exp(-d_i^2/(2\sigma_B^2)). \tag{3.30}$$

and

$$((u_i, v_i))^T \mathbf{\Sigma}_i^{-1}((u_i, v_i)) = \frac{1}{\sigma^2(1 - \rho_i(\sigma_B)^2)} \left(u_i^2 + v_i^2 - 2\rho_i(\sigma_B)u_i v_i\right). \tag{3.31}$$

Then a_B and σ_B^2 can be solved by maximizing the following objective function

$$\sum_{i=1}^{N_p} \log p((u_i, v_i); \kappa_B). \tag{3.32}$$

Here, N_p is the number of pixel pairs. The derivatives of this objective function *w.r.t.* the parameters can be easily derived based on the formulas above.

3.3.2 Learning the Probabilistic Patch Manifold

The model of texture images consists of two modules: the patch manifold and the MRF. As the potential functions of the MRF to enforce coherence across patches are simply the likelihood with respect to the patch manifold model, we only have to estimate the parameters for the patch manifold in the stage of model training.

The patch manifold, as shown in Figure 3-3, involves the following parameters to

be estimated: the prior distribution over components $\boldsymbol{\pi}$, the component parameters $\{(\mathbf{W}_k, \boldsymbol{\mu}_k)\}_{k=1}^K$, and the parameters of the residue distribution α_r and β_r. These parameters together are denoted by $\boldsymbol{\theta}$. This model also involves several latent variables for each patch $\mathbf{y}_c$: the indicator $s_c \in \{1, \ldots, K\}$ that associates it with a component, the latent representation $\mathbf{z}_c$, the orientation ω_c, and the vector of residue variance $\mathbf{v}_c$. These hidden variables together are denoted by $\boldsymbol{\zeta}_c$.

Variational approximation

Direct maximum likelihood estimation of the model parameters $\boldsymbol{\theta}$ is intractable, as it requires integration over all hidden variables. Here, variational EM[3] is employed, which infers the expectation of the hidden variables while optimizing the model parameters. Particularly, we factorize the posterior distribution of these hidden variables into a product as

$$\prod_{c \in \mathcal{C}} q_c(s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c), \tag{3.33}$$

where we approximate q_c as

$$q_c(s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c) = \delta_{\tilde{\omega}_c}(\omega_c) \prod_{j=1}^{d_p} q_{v_c}(v_c^j | \tilde{\alpha}_c^j, \tilde{\beta}_c^j) \sum_{k=1}^{K} \tilde{\pi}_c(k) \delta_k(s_c) \delta_{\tilde{z}_{c,k}}(\mathbf{z}). \tag{3.34}$$

Here, we briefly explain the rationale underlying the choice of this variational approximate:

1. The orientation ω_c has a complex and nonlinear relation with the patch vector $\mathbf{y}_c$. Here, $\delta_{\tilde{\omega}_c}$ is a delta-distribution that assigns probability 1 to $\tilde{\omega}_c$. Thus, in E-steps, the estimation of the variational parameter for ω_c reduces to an optimization problem to find the optimal orientation, which is generally easier to solve. In addition, $\mathrm{E}_{\delta_{\tilde{\omega}_c}}[R(\mathbf{y}_c, -\omega_c)]$ simply degenerates to $R(\mathbf{y}_c, -\tilde{\omega}_c)$, which also makes the M-steps easier.

2. For the residue variance v_c^j, we assume the variational distribution to be an

[3]Section 2.3 provides a brief introduction of the generic variational EM algorithm

inverse gamma distribution $q_{v_c}(v_c^j|\tilde{\alpha}_c^j, \tilde{\beta}_c^j)$. This is a natural choice. Recall that the prior of v_c^j is an inverse gamma distribution, due to conjugacy, the posterior distribution of v_c^j remains in the same family.

3. The choice of the variational distribution for the variables s_c and $\mathbf{z}_c$ is based on the consideration below. The value of $\mathbf{z}_c$ is largely determined by s_c which chooses a specific linear component to explain the patch. Conditioned on $\mathbf{y}_c$ as well as ω_c and $\mathbf{v}_c$, the joint posterior distribution over both s_c and $\mathbf{z}_c$ is a mixture of Gaussian, of which the marginal distribution of $\mathbf{z}_c$ may be multi-modal. Approximating this joint distribution using a product form as $q(s_c)q(\mathbf{z}_c)$ can not properly capture this multi-modal characteristics. The most appropriate form here would be

$$q(s_c, \mathbf{z}_c) = \tilde{\pi}_c(s_c)\mathcal{N}(\mathbf{z}_c|\tilde{\boldsymbol{\mu}}_{s_c}, \tilde{\boldsymbol{\Sigma}}_{s_c}). = \sum_{k=1}^{K} \tilde{\pi}_c(k)\delta_k(s_c)\mathcal{N}(\mathbf{z}_c|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k). \tag{3.35}$$

In our original implementation, this formulation was used, which requires estimating and maintaining K mean vectors and K covariance matrices for each patch during inference. And the inference takes exceedingly long time to run, mostly devoted to the update of $\tilde{\boldsymbol{\Sigma}}_k$. To reduce both time and space complexity, we decided to use a simplified form, which simply replaces the normal distribution with a delta distribution, as

$$q(s_c, \mathbf{z}_c) = \sum_{k=1}^{K} \tilde{\pi}_c(k)\delta_k(s_c)\delta_{\tilde{\mathbf{z}}_k}(\mathbf{z}_c). \tag{3.36}$$

This simplification still preserves the multi-modal characteristics of the marginal of $\mathbf{z}_c$, while substantially reducing the computational cost.

The variational EM steps

With this approximation, the joint objective function of variational EM is

$$J(\boldsymbol{\theta}, \{\boldsymbol{\zeta}_c\}) = \langle \mathrm{E}_{q_c}[\log p(\mathbf{y}_c, s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c|\boldsymbol{\theta}) + H(q_c)] \rangle_c \tag{3.37}$$

Here, $\langle\cdot\rangle_c$ denotes the sample mean over all patches extracted from the training images, as

$$\langle f_c\rangle_c = \frac{\sum_{i=1}^{n}\sum_{c\in\mathcal{C}_i} f_c}{\sum_{i=1}^{n}|\mathcal{C}_i|}. \tag{3.38}$$

Moreover, based on the patch manifold model, we have

$$p(\mathbf{y}_c, s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c|\theta) = p(\mathbf{y}_c|s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c)p(s_c|\boldsymbol{\pi})p(\mathbf{z}_c)p(\omega_c)p(\mathbf{v}_c|\alpha_r, \beta_r). \tag{3.39}$$

The factors in this formula are respectively explained as follows:

1. $p(\mathbf{y}_c|s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c)$ is the conditional pdf of a patch $\mathbf{y}_c$ given the value of latent variables, which is given by

$$p(\mathbf{y}_c|s_c, \mathbf{z}_c, \omega_c, \mathbf{v}_c) = \prod_{j=1}^{d_m}\frac{1}{\sqrt{2\pi(v_c^j)^2}}\exp\left(-\frac{(R(\mathbf{y}_c, -\omega_c)^{(j)} - \mathbf{x}_c^{(j)})^2}{2(v_c^j)^2}\right), \tag{3.40}$$

   where $\mathbf{x}_c = \mathbf{W}_{s_c}\mathbf{z}_c + \boldsymbol{\mu}_{s_c}$.

2. $p(s_c|\boldsymbol{\pi})$ is the prior probability of choosing the component s_c.

3. $p(\mathbf{z}_c)$ follows the standard normal distribution, as given by

$$p(\mathbf{z}_c) = (2\pi)^{-1/2}\exp(-\|\mathbf{z}_c\|^2/2).$$

4. $p(\omega_c) = 1/(2\pi)$, as $\omega_c \sim U([0, 2\pi])$.

5. $p(\mathbf{v}_c|\alpha_r, \beta_r)$ follows the inverse gamma distribution with shape parameter α_r and scale parameter β_r. In particular, it has

$$p(\mathbf{v}_c|\alpha_r, \beta_r) = \prod_{j=1}^{d_p} p(v_c^j|\alpha_r, \beta_r) = \prod_{j=1}^{d_p}\frac{\beta_r^{\alpha_r}}{\Gamma(\alpha_r)}(v_c^j)^{-(\alpha_r+1)}\exp\left(-\frac{\beta_r}{v_c^j}\right). \tag{3.41}$$

Based on the results above, we derive the updating formulas for both E-steps and M-steps. Specifically, the **E-steps** update the parameters of q_c given the model

parameters, as follows

$$\tilde{\pi}_c(k) \propto \pi_k \left(-\frac{1}{2}\|\mathbf{r}_{c,k}\|^2_{\tilde{\mathbf{\Lambda}}_c} - \frac{1}{2}\|\tilde{\mathbf{z}}_{c,k}\|^2 \right); \tag{3.42}$$

$$\tilde{\mathbf{z}}_{c,k} = \left(\mathbf{I} + (\mathbf{W}_k^T)\tilde{\mathbf{\Lambda}}_c \mathbf{W}_k \right)^{-1} ((\mathbf{W}_k)^T \tilde{\mathbf{\Lambda}}_c (R(\mathbf{y}_c, -\omega_c) - \boldsymbol{\mu}_k)); \tag{3.43}$$

$$\tilde{\alpha}_c^j = \alpha_r + \frac{1}{2}; \tag{3.44}$$

$$\tilde{\beta}_c^j = \beta_r + \frac{1}{2} \sum_{k=1}^{K} \tilde{\pi}_c(k) \left(\mathbf{r}_{c,k}^{(j)} \right)^2. \tag{3.45}$$

The **M-steps** update the model parameters given q_c. The updating formulas are:

$$\hat{\boldsymbol{\pi}}(k) = \langle \tilde{\pi}_c^{(k)} \rangle_e; \tag{3.46}$$

$$\hat{\boldsymbol{\mu}}_k = \langle \tilde{\pi}_c^{(k)} \tilde{\mathbf{\Lambda}}_c \rangle_c^{-1} \langle \tilde{\pi}_c^{(k)} \tilde{\mathbf{\Lambda}}_c (\mathbf{y}'_c - \hat{\mathbf{W}}_k \mathbf{z}_{c,k}) \rangle_c; \tag{3.47}$$

$$\hat{\mathbf{W}}_k = \langle \tilde{\pi}_c^{(k)} \tilde{\mathbf{\Lambda}}_c (\mathbf{y}'_c - \hat{\boldsymbol{\mu}}_k) \tilde{\mathbf{z}}_{c,k} \rangle_e \langle \tilde{\pi}_c^{(k)} \tilde{\mathbf{\Lambda}}_c \mathbf{z}_c \mathbf{z}_c^T \rangle_c^{-1}. \tag{3.48}$$

In addition, the scalar parameters α_r and β_r of the inverse gamma distribution can be obtained via MLE over the approximate distribution given by q_{v_c}. Specifically, we can minimize the objective below numerically

$$\sum_{j=1}^{d_p} ((\alpha_r + 1)\bar{\eta}^j + \beta_r \bar{\lambda}^j) - d(\alpha_r \log \beta_r - \log \Gamma(\alpha_r)). \tag{3.49}$$

Here, $\bar{\eta}^j = \langle \tilde{\alpha}_c^j / \tilde{\beta}_c^j \rangle_c$ and $\bar{\lambda}^j = \langle \log(\tilde{\beta}^j) - \psi(\tilde{\alpha}^j) \rangle_c$, where ψ is the digamma function.

Initialization

Using variational EM requires all model parameters to be properly initialized. Here, we describe the specific way that we chose to perform initialization in our experiments. First, we group all patches from all images by K-means into K clusters, where K is empirically set. For each cluster, we apply probabilistic PCA [102] to estimate $\boldsymbol{\mu}_k$ and $\mathbf{W}_k$. After that, we set $\boldsymbol{\pi}$ to be the relative weights of these clusters, and obtain α_r and β_r by performing MLE on the residues. This completes the initialization. Generally, there can be alternative approaches to accomplish this. However, exploring different

initialization schemes is not in the scope of this thesis.

Separate training strategy

The patch manifold is a mixture model. In practice, a divide-and-conquer strategy can be used to estimate the model, that is, learn different sets of components from different data sets, and then put them together into a unified mixture model. This separate training strategy parallelizes the training procedure and reduces memory demands.

We applied this strategy in our experiment. In particular, we group all images in the training set into five categories: *nature*, *animals*, *people*, *buildings*, and *shore*, and respectively learn a patch manifold model for each.

The design parameters are set empirically to balance accuracy and model complexity. In particular, we set the number of mixture hyperplanes to $K = 160$ for each category, and fix their dimension to be $q = 12$. After category-specific models are learned, we combine them into a unified model by simply putting all components together and re-normalizing their prior weights. The unified model is then used to solve the image recovery problems, which we will discuss in next section.

3.4 Application to Image Recovery

We apply the image model to solve low level vision problems, including image denoising and inpainting. Generally, an *observed image* O is given, which is assumed to be generated from an *underlying image* I by a measurement process. Inference of I can be formulated as MAP estimation:

$$\hat{I} = \underset{I}{\operatorname{argmax}} \; p(I|\boldsymbol{\theta})p(O|I;\boldsymbol{\eta}). \tag{3.50}$$

Here, $\boldsymbol{\theta}$ is the parameter of the image prior, and $\boldsymbol{\eta}$ is the parameter of the measurement model. Different low level vision tasks have different measurement processes, which, nonetheless, can be solved with the same image model. This is one significant

advantage of the generative approach.

3.4.1 Image Denoising

We consider a measurement process, where the image is corrupted by Gaussian white noise, as

$$O(x) = I(x) + \varepsilon_x, \quad \text{with } \varepsilon_x \sim \mathcal{N}(0, \sigma_\varepsilon^2). \tag{3.51}$$

Directly solving Eq.(3.50) involves the intractable integration over the latent variables. Again, we resort to variational EM, which is based on the mean field approximation given in (3.33). Here, E-steps update the parameters of q, the approximate posterior of the latent variables, while M-steps update the both the base image B and the texture image Y. (Recall that the underlying image I is modeled as $B + Y$).

The E-steps use the same formulas as those derived for the learning algorithm (see Eq.(3.42) to (3.45)). Here, $\mathbf{y}_c$ are simply a patch of Y, which is known when Y is given. The M-steps estimate B and Y, given q and the model parameters $\boldsymbol{\theta}$. Specifically, given q, we have

$$\mathrm{E}_q\left[\log(Y|\tilde{\mathbf{h}}, \boldsymbol{\theta})\right] = -\sum_{c\in C}\sum_{k=1}^{K} \tilde{\pi}_c(k)\tilde{E}_{c,k}. \tag{3.52}$$

Here, $\tilde{\mathbf{h}}$ denotes the hidden variables associated with all patches. According to Eq.(3.22), we derive the expected energy $\tilde{E}_{c,k}$:

$$\tilde{E}_{c,k} = \frac{1}{2}\|R(\mathbf{y}_c, -\omega_c) - (\boldsymbol{\mu}_k + \mathbf{W}_k\mathbf{z}_{c,k})\|^2_{\tilde{\boldsymbol{\Lambda}}_c}. \tag{3.53}$$

Here,

$$\tilde{\boldsymbol{\Lambda}}_c = \mathrm{diag}((\tilde{\lambda}_c^j)_{j=1}^{d_p}), \qquad \text{and} \qquad \tilde{\lambda}_c^j = \mathrm{E}_q((v_c^j)^{-1}) = \tilde{\alpha}_c^j/\tilde{\beta}_c^j.$$

Eq.(3.52) and (3.53) together leads to a prior energy function over Y that contains only linear and quadratic terms. This is equivalent to imposing a "mean Gaussian MRF" over Y, conditioned on the variational parameters of the hidden variables, which we denote by $\tilde{G}_Y$. Therefore, the inferential M-steps maximize the following

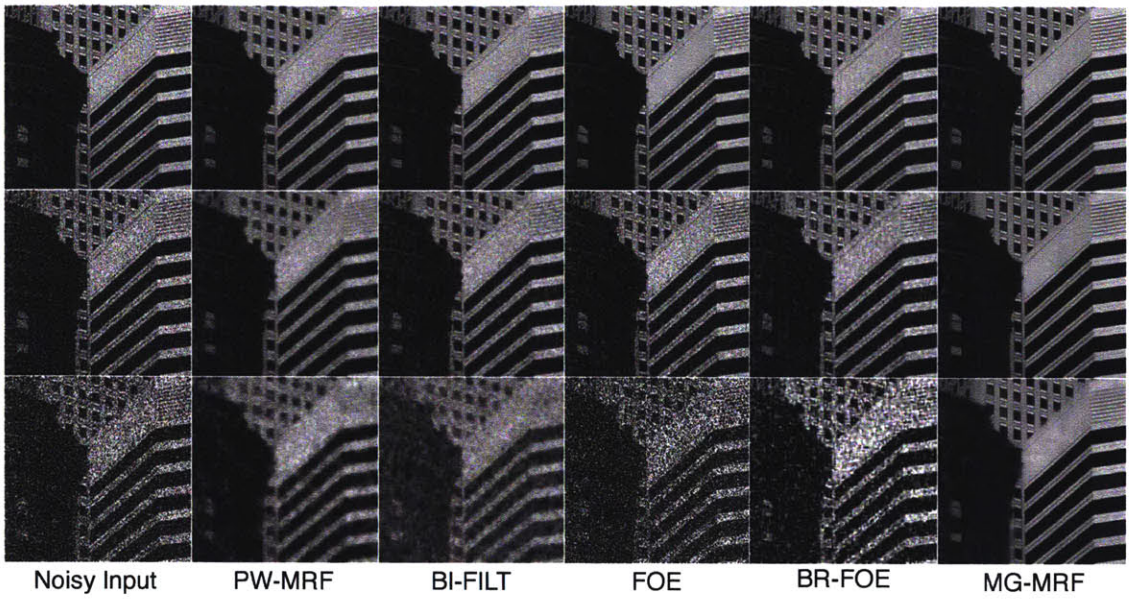


Figure 3-6: The input noisy images (the first column) with the recovered images obtained with different methods. Only part of the images are shown to highlight the differences between methods (see the full clean image in Figure 3-7). The inputs at different rows are subject to different levels of noise ($\sigma = 0.1, 0.2, 0.5$).

function with respect to Y and B:

$$p(Y|\tilde{G}_Y)p(B|G_B)p(O|Y+B), \tag{3.54}$$

Here, $p(B|G_B)$ is the GP-prior of the base image, and $p(O|Y+B)$ is the model given in Eq.(3.51). Particularly, we have

$$p(O|Y+B) \propto \exp\left(-\frac{1}{2}\sigma_\epsilon^{-2}\sum_{x\in\mathcal{D}_I}(Y(x)-B(x)-O(x))^2\right). \tag{3.55}$$

Here, $\mathcal{D}_I$ is the image domain, *i.e.* the set of all pixel coordinates.

It is easy to see that the posterior of B and Y are jointly Gaussian, as all factors above are Gaussian. Hence, given q, the problem reduces to the inference over a Gaussian MRF, which can be readily solved via quadratic programming.

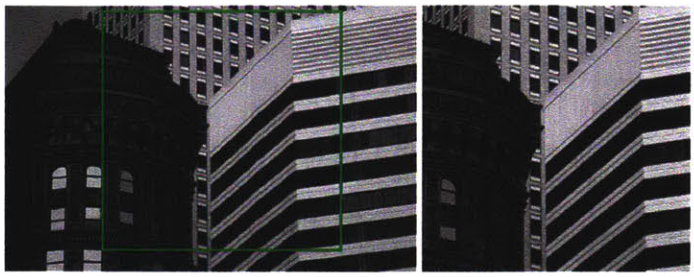


Figure 3-7: The clean image underlying the inputs in Figure 3-6.

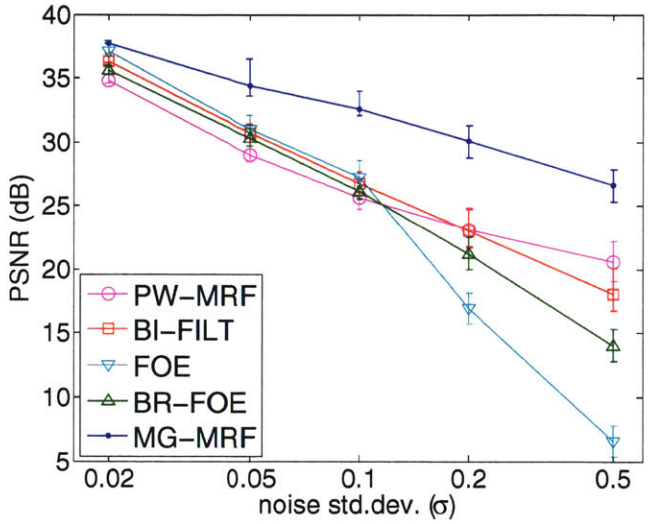


Figure 3-8: Each curve shows the median of the PSNR values on all testing images. The bars below and above each data point are respectively the 25% and 75% quantiles.

Experiment results

In the experiments, we examine the robustness of the method to a range of noise variance. We also compare the proposed method (MG-MRF) with four other methods on image denoising, which include the classic pairwise MRF (PW-MRF), bilateral filtering (BI-FILT) [76], field of experts (FOE) [83], and Weiss's variant of FoE (BR-FOE)[115]. When using MG-MRF for denoising, the MRFs are built upon overlapping patches of size 13×13 with 3-pixel interval. Under this setting, each pixel is covered by 16 to 20 patches, which provides a balance between coherence, robustness, and computational efficiency.

The inference algorithm takes 5 to 30 iterations to converge. In general, more iterations are required under higher noise levels. We implement the algorithms for

Figure 3-9: The clean images underlying the set of additional results.

PW-MRF and BI-FILT, and use the code published by the authors of the corresponding papers for FOE and BR-FOE. Here, the FoE model is constructed with 5×5 cliques and 24 filters. We seek the best settings of design parameters via cross validation for all comparison methods, and evaluate the performance in terms of peak signal-to-noise ratio (PSNR) in dB.

Figure 3-6 shows the denoising results obtained on a test image. The corresponding uncorrupted image are shown in Figure 3-7. Generally, when the noise is moderate ($\sigma = 0.1$), PW-MRF, as expected, tends to slightly blur edges; while other methods preserve edge sharpness. Close examination reveals that the image generated by MG-MRF is qualitatively better than the others. As the noise level increases, MG-MRF continues to perform robustly except for minor blurring of boundaries between different texture patterns; while other methods degrade noticeably. Interestingly, when $\sigma = 0.5$, PW-MRF performs significantly better than both FOE and BR-FOE. This observation is consistent with the dependence of FoE methods on derivative filter responses, which are sensitive to high noise levels.

Figure 3-10 and Figure 3-11 show additional results. The corresponding uncorrupted images are shown in Figure 3-9. In all these tests, the proposed method consistently outperforms others.

Figure 3-8 summarizes the performance statistics obtained over the images in

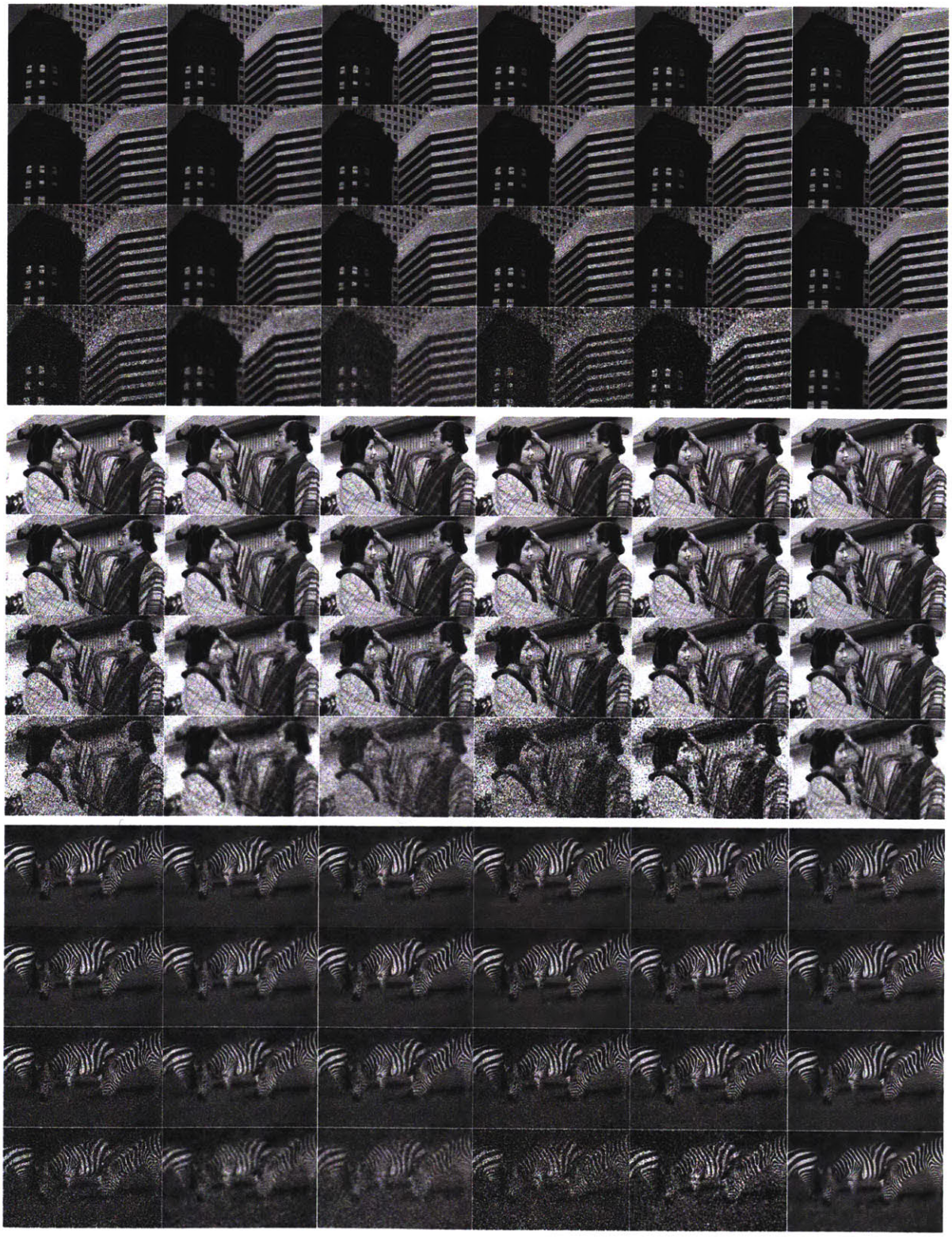


Figure 3-10: The first set of additional results on image denoising. The six columns from left to right respectively show the noisy input, and the results obtained using PW-MRF, BI-FILT, FOE, BR-FOE, and MG-MRF.

Figure 3-11: The second set of additional results on image denoising. The six columns from left to right respectively show the noisy input, and the results obtained using PW-MRF, BI-FILT, FOE, BR-FOE, and MG-MRF.

the test set, under different noise conditions (*i.e.* $\sigma_\varepsilon = 0.02, 0.05, 0.1, 0.2$ and 0.5). In general, the methods based on pairwise links (PW-MRF and BI-FILT) degrade more gracefully than the FoE-based methods (FOE and BR-FOE) as the noise level increases. MG-MRF consistently outperforms other methods.

The experimental results demonstrate that MG-MRF is superior to other methods in two aspects: preservation of texture details and robustness to high noise levels. This is a consequence of its distinctive mechanism in which the oriented templates derived from the learned patch manifold generate local patterns, and are combined with an MRF to ensure coherence between them. This is in contrast to prior methods using MRFs which impose coherence at the pixel level. When the noise variance is large, the direct influence of the observed pixel values becomes insignificant. The inference algorithm uses the observed image mainly for choosing templates from the manifold. Note that each choice is conditioned on all 169 pixels in a patch, making it much more robust than the methods that rely on a much smaller neighborhood. The Bayesian formulation utilizing a distribution of models instead of a single model also contributes to the reliability.

3.4.2 Image Inpainting

The task of inpainting is to recover missing portions of a partially observed image. Suppose we are to recover an image I. Let $\mathcal{O}$ and $\mathcal{U}$ respectively denote the set of observed and missing pixels, and $I(\mathcal{O})$ denote the observed pixel values. The problem here is to infer the value of $I(\mathcal{U})$.

Similarly, we can apply variational E-M to solve this problem, with E-steps updating q, the approximate posterior of the associated latent variables, and M-steps updating the base image B and the texture image Y. Here, the E-steps follow the same formulas as in image denoising, while the M-steps are different. As discussed above, given q, there is a Gaussian Markov random field over Y, denoted by $\tilde{G}_Y$. The M-steps maximize the following function with respect to Y and B:

$$p(Y|\tilde{G}_Y)p(B|G_B) \tag{3.56}$$

under the following constraint

$$Y(\mathcal{O}) + B(\mathcal{O}) = I(\mathcal{O}). \tag{3.57}$$

Instead of solving this constrained optimization problem, we reformulate it as an equivalent unconstrained problem that involves three variables $Y(\mathcal{U}), B(\mathcal{U}), Y(\mathcal{O})$, by replacing $B(\mathcal{O})$ with $I(\mathcal{O}) - Y(\mathcal{O})$. The resultant problem remains a quadratic programming problem (but without constraints), which can be readily solved.

Initialization

Special care should be taken to bootstrap the E-M algorithm. Here, I describe an effective procedure to initialize the variable values.

First, we obtain $B(\mathcal{O})$ by excessively blurring the observed region, and solve $B(\mathcal{U})$ purely based on the prior Gaussian process. This has a close-form solution. Let $\mathbf{b}_o$ and $\mathbf{b}_u$ respectively denote the vector of pixel values in $B(\mathcal{O})$ and $B(\mathcal{U})$, then the optimal value of $\mathbf{b}_u$ is given by

$$\hat{\mathbf{b}}_u = \mu_B + \mathbf{C}_{uo}\mathbf{C}_{oo}^{-1}(\mathbf{b}_o - \mu_B). \tag{3.58}$$

Here, $\mathbf{C}_{uo}$ is the prior covariance matrix between $\mathbf{b}_u$ and $\mathbf{b}_o$ and $\mathbf{C}_{oo}$ is the prior covariance of $\mathbf{b}_o$. Both can be directly derived from the Gaussian process.

Next, we initialize the texture image Y. Here, $Y(\mathcal{O})$ can be easily determined, as $Y(\mathcal{O}) = I(\mathcal{O}) - B(\mathcal{O})$. The part $Y(\mathcal{U})$ can be derived by greedily filling in the missing pixels, from boundary towards the center.

At each iteration, we pick a partially observed patch with the least missing pixels, and evaluate the marginal likelihood of the observed part *w.r.t.* all components of the patch manifold, choosing the one that yield highest value to explain the patch. Then, we infer the optimal values of the missing pixels in this patch using the chosen component (recall that each component is a Gaussian distribution). This process continues until all missing pixels are filled, which provides a reasonably good initialization.

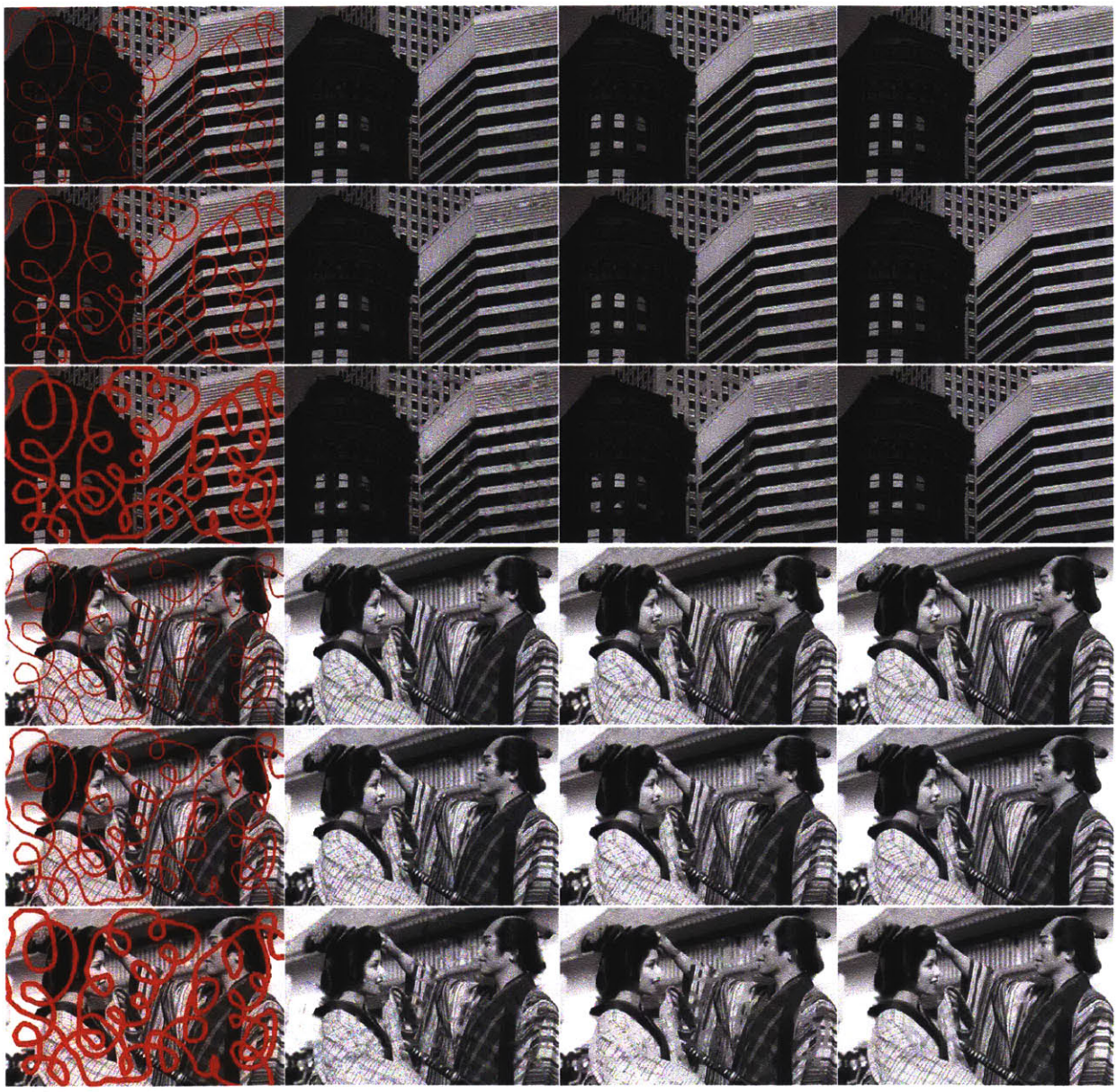


Figure 3-12: The results of inpainting on partially observed images with masks of different widths. From left to right are the masked inputs, and the results obtained using FOE, TV-MRF, and MG-MRF.

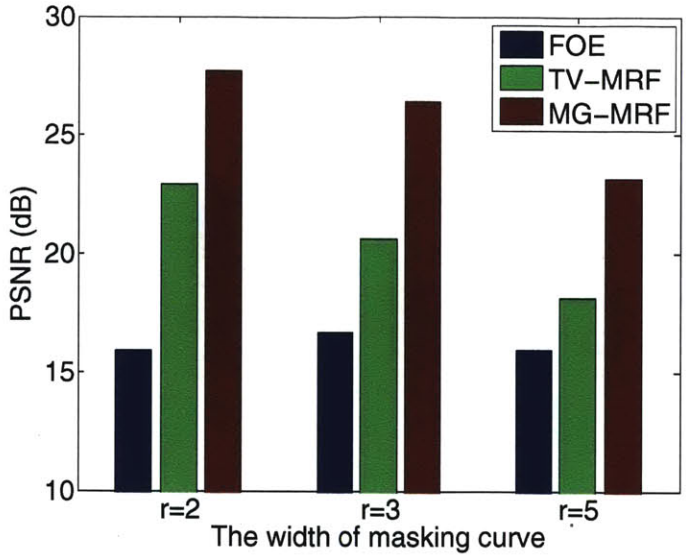


Figure 3-13: The PSNR of inpainting results within masked region.

One can further improve the quality of initialization by simultaneously filling in multiple patches. Specifically, for each patch residing on the boundary of missing and observed region (*i.e.* the ones that contain both observed and missing pixels), we first choose a Gaussian component to explain it as above. All these Gaussian components together constitute a joint Gaussian distribution over the boundary patches, with which all missing pixel values at these patches can be jointly inferred.

Experiment results

Image inpainting is to infer the missing part given a partially observed image. To test the algorithm under different conditions, we generate occlusion masks of different widths. Specifically, we draw a free-form curve as a skeleton, and dilate it to a specific width to generate the mask.

For inpainting, we compare our method with two other MRF-based approaches: the FoE-based method [83] and TV-MRF regularized recovery. The number of iterations needed to recover an image increases as the width of masking curve increases. Figure 3-12 shows results for two example images.

When the mask width is large, the results yielded by both FOE and TV-MRF contain noticeable artifacts (see the third row of each set of results), especially at

places the masking curve passes through complex patterns. While MG-MRF performs better in recovering such patterns, as they are effectively captured by the texture manifold. We also perform quantitative evaluation, in terms of PSNR within the masked region. The results shown in Figure 3-13 show that MG-MRF works better than the comparison methods for all three different mask widths.

3.5 Summary

We developed a generative image model for low level vision, which incorporates a patch manifold to model the local texture patterns, and a conditional MRF to ensure coherence between patches. With a mean field approximation, we derived efficient algorithms for both learning and inference, which we apply to image denoising and inpainting.

The experimental results demonstrate that our method performs substantially better than other methods in recovering complex texture patterns, and shows superior robustness against severe noise corruption. Such improvement is ascribed to the patch model that is more effective than an MRF based on derivative filters in capturing local structures, as well as the Bayesian approach that adaptively combines the MRF predictions in posterior inference.

Chapter 4

The Motion Model

As a key aspect in dynamic scene modeling, *motion* plays a crucial role in a wide variety of vision tasks, such as surveillance, even detection, and video analysis.

While the research on motion analysis has a long history, much of existing work focuses on developing techniques to estimate local velocity, such as object tracking and optical flow. The reliance on local observations (*i.e.* those within a small region and at a particular time step) restricts their capability of resolving many ambiguities arising in practice. Moreover, with the results produced by such methods (*e.g.* pixel-wise velocities or tracks of individual objects), it remains a nontrivial problem to derive a coherent interpretation of the observed motion.

To address these issues, I introduced the notion of *geometric flow* to motion modeling, which provides a higher level formulation that is able to capture common motion patterns over both space and time. On top of geometric flows, a linear representation based on Lie algebra is derived, with which a family of flows can be mapped to a vector space with each flow characterized by a coefficient vector. The Lie algebraic representation greatly simplifies probabilistic modeling of flows. Taking advantage of this, we further formulate a stochastic flow model and apply it to analyze motion in real world videos.

Figure 4-1: This figure shows the frames respectively captured in three different dynamic scenes that exhibit obvious persistent motion patterns: the flow of water in a spring, cars running on a road, and athletes running along a circular path.

4.1 Overview of Motion Models

Modeling and analysis of motion patterns in video is an important topic in computer vision. While extensive efforts have been devoted the problem of local motion estimation, such as tracking individual objects or estimating optical flows between consecutive frames, research on modeling persistent motion patterns has received less attention. Persistent motions are ubiquitous. In many applications, such as scene understanding and crowd surveillance, one is primarily interested in *collective and persistent motion patterns* rather than the motions associated with individual entities. Figure 4-1 depicts frames in three different video sequences. In such scenes, characterizations such as *the vehicles are moving towards bottom right corner with a slight rotation* and *the athletes are running along a circular path* are more pertinent than the velocities of individual objects.

To model persistent motion patterns over both space and time, we explore a new methodology in this work, aiming to develop a new model that is able to leverage the geometric coherence in dynamic motion. Particularly, we introduced a new characterization which describes motion patterns using *geometric flows*, a notion originating from differential geometry.

4.1.1 Review of Related Work

Research on motion analysis has a long history, and numerous algorithms and models have been proposed, which mostly fall into four categories: *tracking*, *optical flows*, *deformable models*, and *space-time features*.

Tracking

A tracking algorithm is employed to keep track of the locations and other states of the interested objects across frames. At each frame, the location of each object is determined via local search around a predicted center. This is often formalized as a Bayesian filtering problem[18][61][71].

Kalman filtering[116] and Particle filtering[18][6] are two most widely used filtering techniques in dynamic analysis. Both incorporate a hidden markov chain to model the transition of object states (such as locations) based on temporal continuity or other kinematic assumptions. An appearance model (*e.g.* a template) is used to connect the internal states to observed image sequences.

In particular, Kalman filtering[116] assumes linear dependencies (in form of conditional Gaussian distribution) between temporally consecutive frames. Taking advantage of the mathematical properties of Gaussian distributions, the inference can be done efficiently using analytic formulas. Particle filtering[18][6] is based on sequential importance sampling. It uses a collection of weighted particles to represent the posterior distribution of states, which evolve over time via resampling or reweighting. Particle filtering is much more flexible than Kalman filtering in handling non-Gaussian cases, however, it tends to be much slower in practice.

In addition to object locations, other features are often incorporated as states during the tracking process. Typical features that are utilized in tracking include statistics of the intensity or colors[20][92], edges[25][26], or their hybrids[71][42].

Efforts devoted to improving the efficiency, robustness, and accuracy of tracking have substantially advanced the the state-of-the-art in the past decade. Latest tracking systems can achieve satisfactory performance in controlled environments.

However, reliable tracking remains a significant challenge under general conditions, where occlusions may occur frequently and objects distant from cameras may be severely blurred.

Optical flow

Optical flow methods use a dense map of local velocities to represent the motion between two consecutive frames. There are two families of optical flow algorithms, respectively originating from the Horn-Schunk method [47] and the Lucas-Kanade method [65].

The Horn-Schunk method [47] is based on the assumption of constant intensity. Through first-order approximation, this results in a linear relation between image gradients and the local velocity. This relation can be used to determine the velocity component along the gradient direction, but not the orthogonal one, leading to the *aperture problem*. To address this issue, smoothness is often enforced to regularize the estimation. The Lucas-Kanade method [65] takes a block-wise approach, where each block is associated with a velocity (or an affine transform) that can be determined through iterative regression. The aperture problem is effectively mitigated by using blocks instead of individual pixels.

Numerous optical flow estimation methods have been developed to make improvements upon the standard algorithms in different aspects. Baker and Anandan [12] proposed to use Markov random fields to enforce smoothness and use a robust energy function instead of the squared error in order to suppress the effect of outliers. Bruhn *et al.* [15] developed a method that combines the Horn-Schunk's smoothness regularizer and Lucas-Kanade's shared estimation strategy to give a smooth and reliable estimation. Weickert *et al.* [112] studies different types of convex regularizers of the flow fields. Ince and Konrad [49] proposed a methodology that simultaneously determines the optical flow and the occlusion, and breaks the smoothness constraint at the places where occlusion occurs. Sun *et al.* [95] introduced a probabilistic model of optical flows, which casts the optical flow estimation problem to a maximum-a-posteriori inference problem. Lefevre *et al.* [60] extended the optical flow formulation

to generic Riemann manifolds. A comprehensive review of these algorithms is beyond the scope of this thesis. Interested readers can refer to Mitiche's review [70] or Baker's comparative study [7].

One drawback that suffered by many optical flow estimation methods is the aperture problem mentioned above, which is rooted in the optical flow equation that serves as the basis of these methods. While smoothness via regularization alleviates this problem, it introduces another issue – blurring across motion boundary. Moreover, the underlying constant-intensity assumption makes the algorithms vulnerable to illumination changes.

Deformable models

This family of models is used to track the dynamic movement and deformation of specific classes of objects. These models keep track of the location and shape of objects, and actively update them over time. Models in this family can be roughly classified into several types as follows.

1. *Contour-based models.* A representative model of this type is Active contour[52]. Typically, it represents a contour as a string of points, and seeks the best contour by minimizing an energy function that balances the tendency of placing the contour near edges and the smoothness of the contour. Shape priors are sometimes incorporated to regularize the solution. The active contour algorithm has been improved by a lot of work[81][75][68] since it is proposed.

2. *Level-set methods.* The level set method[67] is a different technique for tracking curves, which are represented as the zero level set of an auxiliary function. One important advantage of level set representation over explicit contour representation consists in its inherent capability of handling the variation of curve length and change of topology.

3. *deformable templates.* The methods that rely only on boundaries, including active contour and level set methods, neglect the interior contents which would also contain significant information for motion estimation. Actually, models

that integrate both shape and interior appearance have also been developed. Two representative ones include Active shape model[22] and Active appearance model[23][56]. In these models, the shape is captured by a deformable mesh that is iteratively updated to match the deformed template to the observed image.

A recent work called Metamorphs[48] further extends this idea, which unifies the contour energy and the appearance energy, and employs a more sophisticated deformation scheme called Free-form deformation.

4. *articulated models.* Articulated models[58][24][44] are very popular in human body tracking. In these models, the object(e.g. a person) is considered as composed by several components, connected via joints. The interaction between different parts is modeled by a Markov network with geometric constraints.

These models work well in the applications that they are respectively tailored to. The main limitation is that each model is restricted to a particular class of objects. They may also encounter difficulties when the structure of these objects is subject to substantial changes.

Space-time features

Recently, models based on local space-time features have emerged as a popular mean to characterize dynamic scenes. Rather than striving for reliable motion estimation, they attempt to explain the scene through statistical models built upon a large collection of local spatio-temporal features that are much easier to acquire.

Shechtman *et al.* [91] proposed a space-time correlation method, in which the dynamics is described by the space-time gradients within small space-time cubes detected over the video by an interesting way of correlation. Efro *et al.* [30] developed a framework that utilizes local statistics of the optical flow field as descriptors for action classification. Lena *et al.* [37] utilized the properties of the solution to the Poisson equation to extract space-time features such as local space-time saliency and orientations, which are then integrated together to give a description of the action.

An important advantage for these methods is that they circumvent the difficulty of reliable motion estimation. Instead of focusing on the accuracy of individual descriptors, these methods treat the entire set of descriptors as a bag of visual terms, using their statistics to characterize a scene. A drawback of this approach is that spatial relations between features at different parts, which often convey significant information, are not utilized.

4.1.2 Motivation: Problems with Existing Methods

As we can see from the review in previous section that many existing methods are *local* by nature, which are reflected in two aspects:

1. They characterize the dynamics of a scene using velocities or short-time tracks of individual points or objects, and focus on accurate estimation of these local velocities. Generally, they do not pursue higher level representation that brings together such local observations, or consider it as a separate modeling problem.

2. They estimate the velocity of an object or a point only based on spatially and temporally local information. The utilization of the relations between the motions of different objects or points is merely restricted to smoothing and regularization.

Such methodologies may be sufficient when one is dealing with a simple scene where the appearance of the moving objects can be clearly seen, and their trajectories can be easily identified. However, the local nature of these methods severely limits their capability of modeling complex dynamic scenes or the scenes captured under adverse conditions. A typical example, in which conventional approaches may encounter difficulties, is the video surveillance of a public area where the people are monitored by a far-field camera with low resolution and low signal-to-noise ratio, and occlusion occurs frequently. Under such circumstance, it is very difficult to accomplish persistent and robust estimation relying only on local information.

In order to address this issue, we should extend our perspective to a broader scope. In a real scene, the dynamic behaviors often exhibit strong coherence within

a region and during a period, which, we believe, can be captured jointly with a unified formulation. Such coherence, if leveraged properly, may lead to two significant advantages:

1. The inherent coherence of a dynamic model connects objects or points at different locations, thus offering a mechanism to share statistical strength and help to tackle ambiguities that would otherwise be difficult to resolve.

2. As common motion patterns are often reflected via a large collection of observations over space and time, they can be estimated more reliably than local descriptions such as the velocities of individual objects.

3. The common behaviors or relations shared by a group of objects or reflected over a large region often convey significant information for higher level analysis, such as interpreting observed phenomena and predicting future evolution.

4.1.3 A New Approach based on Geometric Flows

In this work, our primary goal is to *characterize coherent motion patterns with a unified formulation while preserving flexibility to express natural variations.* Direct application of existing techniques to accomplish this task is challenging. With a narrow focus on temporally and spatially local motion (*e.g.* the velocities of a particular object or at a particular time), current methods do not provide a natural mechanism to aggregate potentially sparse observations over space and time into a unified model. While regularization techniques, such as enforcing smoothness via a Markov random fields, may help to improve the robustness of local estimates, they do not change the way that most exiting methods characterize motion – using local velocities.

Achieving the goal above requires a new representation – a representation devised with broader perspective. Before introducing our new approach, we first review two conventional ways to describe motion. The first is to represent the motion of an object by its *trajectory*, *i.e.* the position of an object or a point as a function of time. Alternately, one might use a *geometric transform* to describe how a region evolves.

While this captures the common behavior of an entire region, it only does so within a small temporal window.

Trajectories characterize motion over time while geometric transforms over space. This motivates the idea to establish a temporally and spatially global representation that unifies trajectories and geometric transforms. Consequently, we introduce the notion of *geometric flow*, which characterizes motion over both space and time, as a unification of a collection of trajectories driven by common rules and a continuous geometric transformation process. Note here that the term *geometric flow* has a precise meaning in differential geometry and that our use is consistent with it.

Analysis later will reveal that a geometric flow can be represented using a velocity field. Nonetheless, it is fundamentally different from an optical flow. Geometric flows describe motion by a continuous geometric transform process, while optical flow represents motion as a dense velocity map where each velocity is estimated locally. Each family of geometric flows is associated with a Lie algebra, *i.e.* a vector space comprised of infinitesimal generators, with each flow represented by a vector in this space. A Lie algebraic representation makes it possible to decompose a flow into a linear combination of base flows, thus greatly simplifying statistical modeling and estimation.

In reality, the trajectory of an object can deviate from the path predicted by the driving flow for a variety of reasons. To account for such uncertainties, a generative stochastic model of flows is formulated, which incorporate a Gaussian process as a prior on the flow parameters so as to capture global coherence more effectively. The stochastic formulation is then generalized to admit multiple concurrent flows by introducing an MRF for flow association. The estimation under this model can be done efficiently using variational EM.

The main contributions of this work are summarized as follows:

1. Introduce the notion of geometric flows to model persistent motion patterns, which unifies trajectories and geometric transforms through their intrinsic connections.

2. Derive a Lie algebraic representation, such that each family of flows can be characterized by a set of basis and thus each flow by a coefficient vector. This greatly simplifies the modeling of flows.

3. Develop specific constructions of parametric family of affine flows, which include affine flows and multi-scale extensions that combine multiple locally affine flows to express complex motion patterns while maintaining global consistency.

4. Formulate a stochastic flow model, which provides a uniform mechanism to integrate different types of observations for robust motion estimation. Standard inference techniques such as variational E-M can then be applied to estimate flow coefficients from noisy observations.

4.2 Geometric Flows

As discussed in previous section, to derive a motion model that can effectively capture coherent motion patterns over space and time, we propose to use *geometric flows*.

4.2.1 The Concept of Geometric Flow

The concept of *flow* that we are going to discuss in this chapter originates from the theory of differential geometry. To distinguish it from other flows in computer vision (*e.g.* optical flow), we call it *geometric flow*. Rather than describing the dynamics as the velocities of individual objects or points, each flow characterizes the motion over a spatial region and a time range. Here, we first review two primary representations used for motion description in previous work:

1. *Trajectory-based descriptions*, often used in person or vehicle tracking systems, collect the kinematic state of an individual object over time, typically independent of other objects in the scene.

2. *Geometric transforms*, often used in object alignment and image registration applications, describe the transformation of points over an entire region.

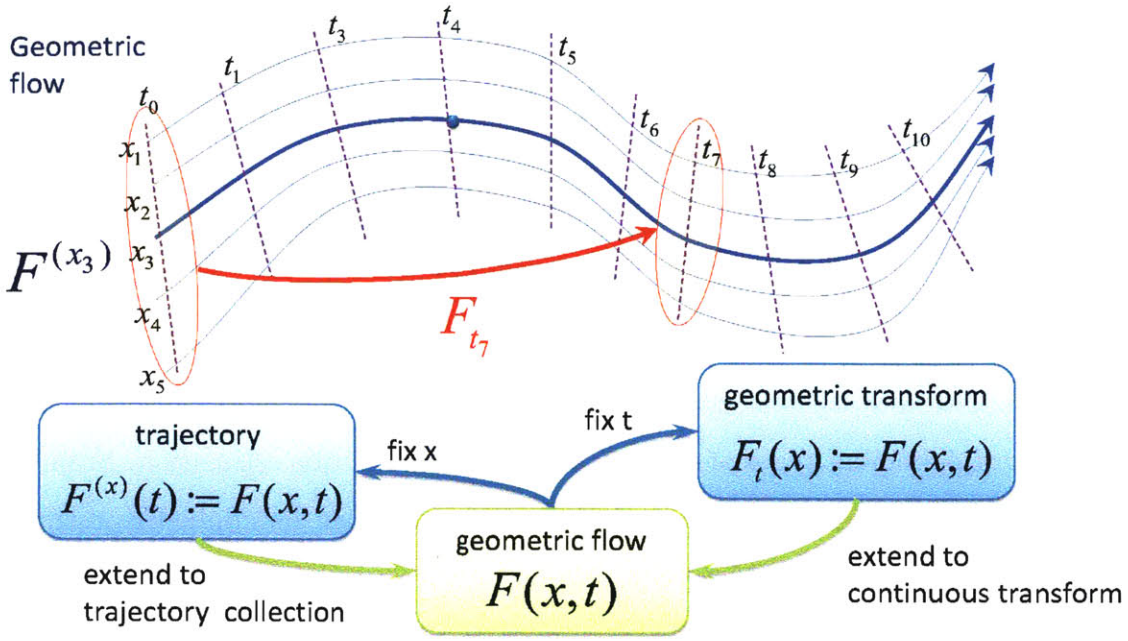


Figure 4-2: Conceptually, a flow can be obtained in either of the following two ways: (1) By inspecting the full motion of a collection of points whose initial locations differ, we get a set of trajectories, or (2) By integrating the geometric transforms terminating at different times t, we get a continuous transform process, which describes how every point within a domain moves over time. In this sense, geometric flows unify trajectory sets and continuous geometric transforms. Conversely, from a flow one can derive the trajectory starting at x, defined by $F^{(x)}(t) := F(x,t)$ or a geometric transform terminated at time t, defined by $F_t(x) := F(x,t)$.

A trajectory describes the motion of a single point over a long time duration, while a geometric transformation describes the motion of all points over a spatial region, but only over a short time window. Although useful for many applications, both representations are lacking when used for modeling coherent motion patterns as neither simultaneously describes motion over *both space and time.*

A geometric flow unifies the descriptions above. Formally, a ***geometric flow*** is defined to be a function $F : \mathbb{R} \times \mathcal{X} \to \mathcal{X}$ that characterizes the motion over a region, which we call the *domain* of F. Here, $\mathcal{X}$ is the image domain (which is $\mathbb{R}^2$ in two-dimensional Euclidean space). Given the initial position $x \in \mathcal{X}$ and time duration t, F yields the destination location at time t. A geometric flow must satisfy two

identities:

$$F(x,0) = x, \quad \forall x \in \mathcal{X}, \tag{4.1}$$

$$F(F(x,t_1),t_2) = F(x,t_1+t_2), \quad \forall x \in \mathcal{X}, t_1, t_2 \in \mathbb{R}. \tag{4.2}$$

Consider a physical point driven by a flow F. Eq.(4.1) simply states that at time $t = 0$ the point is at its initial position while Eq.(4.2) states that geometric flows are *associative*, *i.e.* a point moving along the flow for time t_1 and then for time t_2 is equivalent to moving for time t_1+t_2. Note that t can be negative, allowing "backward tracing". Figure 4-2 illustrates a geometric flow and its relations with trajectories and geometric transforms.

Varying time t over $\mathbb{R}$ yields a family of geometric transforms $\{F_t | t \in \mathbb{R}\}$. It can be shown from Eq.(4.1) and (4.2) that they constitute a one-parameter transformation group isomorphic to the addition group $(\mathbb{R}, +)$, which is the algebraic characterization of a flow. The action of this group on a particular point x leads to the orbit $\{F_t(x) | t \in \mathbb{R}\}$, which is exactly the trajectory that the point would traverse. This analysis makes the intrinsic link between the trajectories and geometric transforms induced by the same flow explicitly.

4.2.2 Lie Group and Lie Algebra

The notion of geometric flow is closely related to the theory of Lie group and Lie algebra. We will see later that by exploiting the intrinsic connections between Lie group and Lie algebra, geometric flows can be mapped to vectors in a linear space, thus leading to a vector representation of flows. To lay the theoretical basis of later discussion, the remaining part of this section will temporarily digress from the main theme of motion modeling and provide a brief and abstract review of the concepts of *Lie group* and *Lie algebra*.

The exposition of the Lie group and Lie algebra theory rests on the basic concepts of group theory and differential geometry (One may refer to Appendix A and B for a brief summary).

Lie Groups

Lie group theory is a beautiful theory where the group theory and manifold theory meet each other. In this theory, the key concept is *Lie group*, which is formally defined below.

Definition 4.1 (Lie Group). *A* Lie group *is a smooth manifold G together with a product operation, such that it is also a group, in which the product operation and the inverse operation are smooth maps, (it is equivalent to that $(g, h) \to gh^{-1}$ is smooth).*

The notion of Lie group subsumes a variety of mathematical entities. For example, non-zero real numbers $\mathbb{R}^*$ with multiplication, positive real numbers $\mathbb{R}^+$ with multiplication, and invertible matrices with matrix multiplication (general linear group $\mathrm{GL}(n, \mathbb{R})$, are all Lie groups. In addition, any direct product (in algebraic sense) of Lie groups remains a Lie group.

A Lie group can have sub-structures, called *Lie subgroups*, as defined below.

Definition 4.2 (Lie Subgroup). *A Lie subgroup of a Lie group G is a subgroup of G together with a smooth structure that makes it an immersed sub-manifold of G.*

One can define functions that map from a Lie group to another. If such a function preserves group structure, it is called a *group homomorphism.* In addition, we have

Definition 4.3 (Lie group Homomorphism). *Let G and H be Lie groups, a smooth map $F : G \to H$ which is also a group homomorphism is called a Lie group homomorphism.*

Definition 4.4 (Lie group isomorphism). *A diffeomorphism that is also a group isomorphism is called a Lie group isomorphism. Any bijective Lie group homomorphism is a Lie group isomorphism.*

Lie Algebra

Each Lie group is associated with a vector space with special algebraic structure, called a *Lie algebra.* This is an important concept, which we rely on to establish

vector space representation of geometric flows. Formally, a *Lie algebra* is defined as follows.

Definition 4.5 (Lie Algebra). *A Lie algebra is a real vector space* $\mathfrak{g}$ *with a binary operation, notated as* $[x, y]$*, called* bracket *operation, which satisfies the following properties:*

1. (Bilinearity): *for all* $x, y, z \in \mathfrak{g}$ *and* $a, b \in \mathbb{R}$,

$$[ax + by, z] = a[x, z] + b[y, z], \tag{4.3}$$

$$[z, ax + by] = a[z, x] + b[z, y]. \tag{4.4}$$

2. (Alternating property): *for all* $x \in \mathfrak{g}$,

$$[x, x] = 0. \tag{4.5}$$

Combination of this with the bilinearity immediately leads to anti-commutativity *as*

$$[x, y] = -[y, x]. \tag{4.6}$$

3. (The Jacobi identity): *for all* $x, y, z \in \mathfrak{g}$,

$$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0. \tag{4.7}$$

Similar to other algebraic structures, an Lie algebra may has its own sub structure, called *Lie subalgebra*, as defined below.

Definition 4.6 (Lie subalgebra). *A subspace* $\mathfrak{h}$ *of a Lie algebra* $\mathfrak{g}$ *is called a Lie subalgebra, if* $\mathfrak{h}$ *is closed under bracket operation.*

Definition 4.7 (Lie algebra homomorphism). *Let* $\mathfrak{g}$ *and* $\mathfrak{h}$ *be Lie algebras, a linear map* $F : \mathfrak{g} \to \mathfrak{h}$ *is called a Lie algebra homomorphism if it also preserves bracket operations*

$$F([X, Y]) = [F(X), F(Y)].$$

An invertible (or equivalently bijective) Lie algebra homomorphism is called a Lie algebra isomorphism.

It is easy to see that the kernel and range of Lie algebra homomorphism are Lie algebras. In addition, the space of $n \times n$ matrices with *commutator bracket* defined by

$$[\mathbf{A}, \mathbf{B}] = \mathbf{AB} - \mathbf{BA}$$

is also a Lie algebra, called the *matrix Lie algebra.*

Relations between Lie Group and Lie Algebra

Lie group and Lie algebra are closely related. In general, the relations between them can be established through *left-invariant vector fields*, *i.e.* vector fields that are invariant to *left translation*:

Definition 4.8 (Left translation). *Let G be a Lie group, any $g \in G$ defines a map $L_g : G \to G$ called left translation as*

$$\forall x \in G,\ L_g(h) = g \cdot x.$$

For any $g \in G$, left translation is a diffeomorphism, whose inverse map is given by $L_{g^{-1}}$.

Definition 4.9 (Left-invariant Vector Field). *A vector field V of a Lie group G is called left-invariant if it is invariant under all left-translations.*

$$\forall g, x \in G,\quad (L_g)_* V(x) = V(gx).$$

Let G be a Lie group, it can be proved that the set of all left-invariant vector fields with bracket operations defined by

$$[X, Y] = X \circ Y - Y \circ X$$

constitutes a Lie algebra, which we called the *Lie algebra associated with* G, denoted by $\mathrm{Lie}(G)$. Note the Lie algebra associated with a Lie group is unique.

The Lie algebra associated with a Lie group G is isomorphic to the tangent space of G at the identity element, as stated by the following theorem.

Theorem 4.1. *Let G be a Lie group. The evaluation map $\varepsilon : \mathrm{Lie}(G) \to T_eG$, given by $\varepsilon(X) = X(e)$ is a vector space isomorphism. Hence,*

$$\dim \mathrm{Lie}(G) = \dim G. \tag{4.8}$$

In addition to the relations between Lie group and its associated Lie algebra, there also exists close relations between their sub structures, as stated by the following theorem.

Theorem 4.2. *Let G be a Lie group, $\mathfrak{g}$ be its associated Lie algebra. H be a Lie subgroup of G, then the Lie algebra associated with H is isomorphic to a Lie subalgebra of $\mathfrak{g}$.*

4.2.3 Lie Algebraic Representation

Traditionally, a given geometric transform can be represented as an element in a Lie group. From the standpoint of statistical modeling, this Lie group-based representation is difficult to work with. The main problem stems from the multiplicative nature of the group structure, which does not support linear operations (addition and scalar multiplication) and thus complicates the application of many statistical learning and inference techniques formulated based on vector spaces.

This issue can be addressed by exploiting the intrinsic connections between the Lie group and the Lie algebra, as follows:

1. Every Lie group G is uniquely associated with a Lie algebra, denoted by $\mathrm{Lie}(G)$, which is a vector space isomorphic to the tangent space at the identity of the Lie group.

2. Each vector in the Lie algebra Lie(G) corresponds uniquely to an element in G via the *exponentiation mapping*: exp : Lie(G) $\rightarrow G$.

3. There exists a neighborhood of the identity element of G, within which every element is uniquely associated with a corresponding element in Lie(G), called the *Lie algebraic representation.*

In general, a Lie algebraic representation has two advantages:

1. The functional form F of a geometric flow is in general nonlinear. As many statistical models presume an underlying vector space, this complicates a statistical model of flows. Exploiting the linear nature of the infinitesimal generator, the Lie algebraic representation largely overcomes such difficulties.

2. Geometric constraints of a flow, which typically restrict the induced transforms to a particular subgroup, are often nonlinear in functional form. Such constraints become linear with the Lie algebraic representation as each subgroup of transforms is described by a linear subspace of the Lie algebra.

4.2.4 Lie Algebra of Affine Transforms

To illustrate the use of Lie algebra in practice, we take the group of affine transformations as an example and show how the Lie algebraic representation benefits the modeling and analysis of geometric transforms. Affine transforms, parameterized by $\mathbf{A}$ and $\mathbf{b}$, have the following form:

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}. \tag{4.9}$$

and can be expressed in homogeneous coordinates as

$$\bar{\mathbf{x}}' = \begin{bmatrix} \mathbf{x}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{T}\bar{\mathbf{x}}. \tag{4.10}$$

While this augmented matrix representation widely used in many vision applications, its use in statistical methods presents some difficulties. First, a group of affine ma-

trices is not a vector space, and thus is not closed under vector addition nor scalar multiplication, complicating the use of statistical learning methods with implicit vector space assumptions. Moreover, it is often the case that one would like to impose geometric constraints upon the transformation. For example, restriction to volume-preserving deformations corresponds to a determinant constraint, i.e. $\det(\mathbf{T}) = 1$. This and a variety of geometric constraints are nonlinear and can be difficult to incorporate into statistical models. The difficulty essentially arises from the fact that the affine group has a multiplicative rather than additive structure. It is desirable to establish a mapping from the multiplicative structure to an *equivalent* vector space representation. This is precisely what the Lie algebra accomplishes in a local sense.

The Lie algebraic representation of a 2D affine transform is a 3×3 matrix with all zeroes on the bottom row. It is related to the homogeneous matrix representation through matrix exponentiation and the matrix logarithm. If $\mathbf{X}$ denotes the Lie algebra representation of $\mathbf{T}$, then

$$\mathbf{T} = \exp(\mathbf{X}) \triangleq \mathbf{I} + \sum_{k=1}^{\infty} \frac{1}{k!}\mathbf{X}^k, \tag{4.11}$$

$$\mathbf{X} = \log(\mathbf{T}) \triangleq \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}(\mathbf{T} - \mathbf{I})^k. \tag{4.12}$$

One advantage of the Lie algebraic representation is that transformation *subgroups* are mapped to linear *subspaces*. Within the 2D affine group, there are many subgroups that correspond to particular families of transforms. This gives rise to a linear parameterization of them. Consider rotations by an angle θ of which the transform matrix $\mathbf{T}_{R(\theta)}$ and the corresponding Lie algebraic representation $\mathbf{X}_{R(\theta)}$ are

$$\mathbf{T}_{R(\theta)} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{X}_{R(\theta)} = \begin{bmatrix} 0 & -\theta & 0 \\ \theta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.13}$$

It can be easily seen that the Lie algebraic representation of all rotations lies in a one dimensional subspace. Similarly, the Lie algebraic representations of many other im-

portant transforms such as scaling, shearing, and translation, correspond to subspaces of the Lie algebra, as well. This property in turn allows for linear characterization of a variety of geometric constraints. Consider the volume-preserving constraint discussed above. Since the composition of two volume-preserving transforms is also volume-preserving. All volume-preserving transformations constitute a subgroup of the affine group. Consequently, their Lie algebraic representations form a subspace. The associated constraint is captured by the simple expression

$$\mathrm{tr}(\mathbf{X}) = 0 \quad \Leftrightarrow \quad X_{11} + X_{22} = 0. \tag{4.14}$$

Here, we just briefly discuss the Lie algebraic representation of affine transforms. Appendix C provides a more detailed study of the affine transformation group and its subgroups, as well as the Lie algebraic characterization of the affine group.

4.3 The Vector Space of Flows

We have discussed the connection between geometric transforms and Lie algebraic representations above. Next, we leverage this connection to derive the Lie algebraic representation of a geometric flow, which is a continuous transformation process instead of a single transformation. As a consequence of this development, we also establish a vector space of flows.

4.3.1 Infinitesimal Generators of Flows

Consider a point driven by a geometric flow F that starts at y and suppose it passes x at time t, i.e. $F^{(y)}(t) = x$. The velocity of the point at t can be obtained by taking the derivative of $F^{(y)}$. A geometric flow has an important property with regards to velocity: Given any x in the flow domain, any point driven by the flow passes through x with the same velocity independent of its initial location. The property implies that each geometric flow F induces a time-invariant velocity field, denoted by V_F, which

can be expressed by

$$\frac{\partial F(x,t)}{\partial t} = V_F(F(x,t)). \tag{4.15}$$

Alternately, given a velocity field V_F, one can reconstruct the flow F by solving the differential equation in Eq.(4.15). This is equivalent to the process of generating the trajectories with the velocities specified by V_F. The *Fundamental Theorem of Flows* [59] states that under mild conditions, each velocity field induces a unique geometric flow:

Theorem 4.3. *Given a smooth flow F on a manifold X, there exists a unique smooth vector field V_F on X such that*

$$\frac{\partial F(x,t)}{\partial t} = V_F(x), \quad \forall x \in X \tag{4.16}$$

Conversely, given a smooth vector field V_F on X, there exists a unique smooth flow F on X with the above equation established.

Consider a transform $F_{\Delta t}$ derived from a flow F. As it induces motion at each point along the velocity given by $V_F(x)$, we have

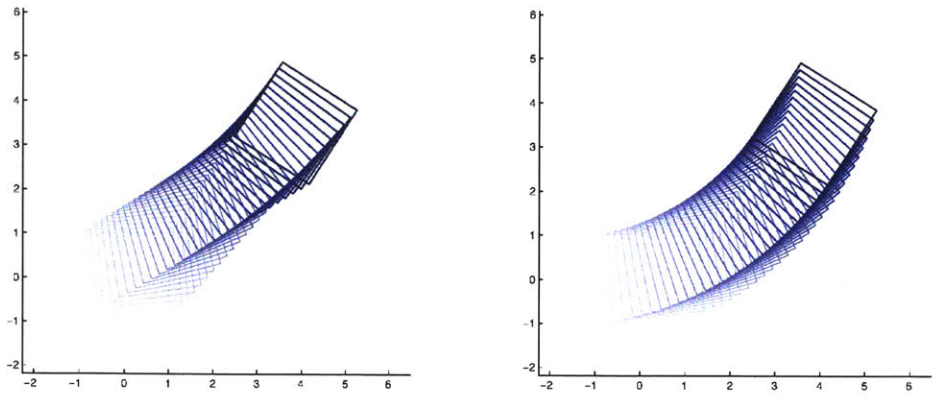
$$F_{\Delta t}(x) \simeq T_{V,\Delta t} := x + V_F(x)\Delta t, \tag{4.17}$$

when the time interval Δt is sufficiently small. We can express each derived transform F_t as a composition of many short time transforms as $F_t = F_{\Delta t} \circ \cdots \circ F_{\Delta t}$. Taking the limit as $\Delta t \to 0$ results in the following equation:

$$F_t = \lim_{N\to\infty} (T_{V_F,\frac{t}{N}})^N. \tag{4.18}$$

This result connects geometric transforms to the driving velocity field. Intuitively, it reflects the observation that *a geometric transform is formed by accumulating the small movements along the underlying velocity field.* Hence, the velocity field V_F is often called the *infinitesimal generator* of the geometric flow F.

In fact, this infinitesimal generator is a generalization of the Lie algebraic repre-



$\mathbf{x}(t) = ((1-t)\mathbf{I} + t\mathbf{T})\mathbf{x}(0)$ $\quad$ $\mathbf{x}(t) = \exp(t\mathbf{X})\mathbf{x}(t)$

Figure 4-3: This figure compares two ways to interpolate transforms to generate a continuous transformation process. The left shows the resultant process obtained using linear interpolation, and the right shows the result obtained using Lie algebra-based interpolation.

sentation introduced above. To see this, let's consider an affine transform $\mathbf{T}$, and try to extend it to a flow F, *i.e.* a continuous transform process, such that $F_1 = \mathbf{T}$, and that for each time $t \in \mathbb{R}$, F_t remains affine. The solution to this problem is unique, which is given by

$$F(x,t) = \exp(t\mathbf{X})x. \tag{4.19}$$

Here, $\mathbf{X}$ is the Lie algebraic representation of $\mathbf{T}$, which plays a key role in extending a transform $\mathbf{T}$ to a continuous process F. Figure 4-3 illustrates the resultant process and compares it with the results generated simply using linear interpolation. In the process resulted from the Lie algebraic construction as above, geometric properties of the shapes are preserved, while the linear interpolation fails to do so.

Taking partial derivatives of Eq.(4.19) with respect to t, we get

$$\frac{\partial F(x,t)}{\partial t} = \frac{\partial}{\partial t}\exp(t\mathbf{X})x = \mathbf{X}x. \tag{4.20}$$

Comparing Eq.(4.16) and Eq.(4.20), we can see that the velocity field V_F plays essentially the same role as $\mathbf{X}$. In particular, then V_F is linear, the derived flow becomes affine. Therefore, V_F can be considered as a generalized Lie algebraic representation

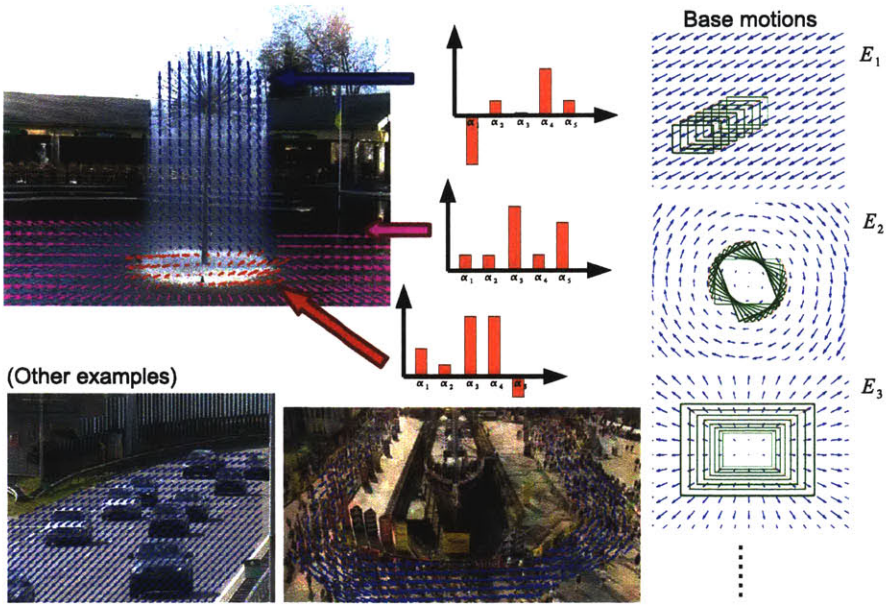


Figure 4-4: This figure demonstrates the representation of a geometric flow as a combination of multiple base flows.

for generic transforms (and thus flows), and correspondingly the *exponentiation mapping* is generalized to be the unique mapping from the velocity field V_F to the induced flow F, as

$$F(x,t) = \exp(tV_F)x. \tag{4.21}$$

Given a group G and its associated Lie algebra $\text{Lie}(G)$, we can construct *a family of flows* $\mathcal{F}_G$ by extending each element in G, as

$$\mathcal{F}_G = \{F : F(x,t) = \exp(tV)x,\ \forall V \in \text{Lie}(G)\}. \tag{4.22}$$

Here, $\text{Lie}(G)$ is also called the *Lie algebra* associated with this flow family $\mathcal{F}_G$, and V is called the *Lie algebraic representation* of the flow F.

Suppose $\text{Lie}(G)$ is an L-dimensional vector space. Given a basis $(E_1, \ldots, E_L)$, the Lie algebraic representation of each flow in the corresponding family can be decomposed into a linear combination of the basis and uniquely characterized by a coefficient vector $\boldsymbol{\alpha} = [\alpha^1, \ldots, \alpha^L]^T$, as

$$V_F = \sum_{l=1}^{L} \alpha^l E_l. \tag{4.23}$$

The vector $\boldsymbol{\alpha}$ is called the ***Lie algebraic coefficients*** of the flow F with respect to the given basis.

Thus far we have developed a vector space representation of flows, and as a consequence, we can express each flow in a family as a combination of the base flows, as illustrated in Figure 4-4. The Lie algebra plays a crucial role in this development.

4.3.2 Flow Actions

In practical applications, the locations of points are often not available. Traditionally, one may rely on tracking or optical flow estimation techniques to derive the trajectories of points, which are often not reliable enough. In this work, we develop a new approach that can directly infer the flows from the changes of images, without the need of deriving local velocities. This approach is based on the notion of *group action* and *flow action.*

Basic concept of group action

Again, we digress temporarily to review the concept of *group action* as a theoretical preparation for later introduction of an important concept – *flow action.*

Definition 4.10 (Group Action). *Given a group G and a set X, a binary operation $\cdot : G \times X \to X$ is called a* group action *of G on X, if it satisfies the following properties*

1. $e \cdot x = x, \ \forall x \in X$, *where e is the identity of the group G;*
2. $g_2 \cdot (g_1 \cdot x) = (g_2 g_1) \cdot x, \ \forall g_1, g_2 \in G, \ x \in X.$

Here, X is called a G-space.

There are several related concepts that are useful in characterizing an action:

1. Given $x \in X$, the *orbit* containing x is defined to be the set $G \cdot x = \{g \cdot x | g \in G\}$, which comprises all "transformed version" of x yielded by the group G.

2. Given $x \in X$, then $g \in G$ is called a *stabilizer* of x, if it does not change x, i.e. $g \cdot x = x$. It can be easily seen that all stabilizer of x constitute a subgroup of G, which is called the *isotropy group* of x.

3. The action of G on X is called a *free action*, if for each x we have $g \cdot x = x \Rightarrow g = e$, which means that only identity element can keep a point unchanged. This is equivalent to that the isotropy group of every x is $\{e\}$. Under such condition, we say that G acts on X *freely*.

If G is a Lie group, X is a smooth manifold, and each element $g \in G$ acts on X smoothly, then the action as defined above is called a *Lie group action*. In this work, we focuses on Lie group actions. For Lie group action, there is an important result in Lie group theory:

Theorem 4.4. *Let G be a Lie group action that acts on X properly and freely, then for each $x \in X$, then the orbit $G \cdot x$ is diffeomorphic to G.*

This indicates that the algebraic and topological structure of a group is completely characterized by each of the orbits that it yields, making it possible to make inference of the transform based on the orbits.

Extension to flow action

Next, we further extend *group action* to *flow action*. Recall that each flow F is a continuous transform process. Therefore, we can define the action of flow F on a space X to be a map that sends each initial location $x \in X$ to an entire trajectory $F \cdot x$, as follows

$$(F \cdot x)(t) := F_t(x) = F(t, x). \tag{4.24}$$

Let V_F be the Lie algebraic representation of F, which lies in a Lie algebraic space with bases $E_1, \ldots, E_n$. Then, we can write V as a linear combination of the bases, as $V_F = \sum_{l=1}^{L} \alpha^l E_l$. As a result, we have

$$(F \cdot x)(t) = \exp(-tV_F) \cdot x = \exp\left(-t\left(\sum_{i=1}^{L} \alpha^l \, E_l\right)\right) \cdot x. \tag{4.25}$$

Figure 4-5: The illustration of the relation between the decomposition of flows and the decomposition of the changes along the image orbit.

This equation is in a nonlinear form that is difficult to work with. To address this issue, we first introduce *the action of infinitesimal generator* V_F on x, which is defined to be

$$(V \cdot x) = \left.\frac{\partial}{\partial t}\right|_{t=0} (F \cdot x)(t) = \left.\frac{\partial}{\partial t}\right|_{t=0} F(x,t). \tag{4.26}$$

Based on this definition and the representation of V as a linear combination, we further obtain

$$(V \cdot x) = \sum_{l=1}^{L} \alpha^l (E_l \cdot x) = \sum_{l=1}^{L} \alpha^l \left.\frac{\partial}{\partial t}\right|_{t=0} \exp(tE_l) \cdot x. \tag{4.27}$$

This result establishes the isomorphism between the Lie algebraic representations and the results of their action on X, which is the theoretical foundation of our inference approach. Intuitively, the infinitesimal change generated by the flow F can be decomposed as a linear combination of the "base changes" $(E_l \cdot x)$ in the same way as the decomposition of the flow itself into combination of base flows. Therefore, the inference problem reduces to a regression problem on the the space of infinitesimal changes.

The space X to be acted on can be a space of any available observations, such as point locations, images, and contour curves, etc. The linear relation established in Eq.(4.27) can be utilized to directly connect the flow coefficients to the dynamic changes of observations. Figure 4-5 gives an illustration of this notion, when the flow is considered as directly acting on the image space.

4.3.3 Multi-scale Extensions

The dynamics in a real scene may exhibit different characteristics in different scales. In a complex scene, while there exist common patterns that can be captured by a global flow, different local parts may have different local characteristics respectively. Hence, complete modeling of such dynamic scenes calls for a multi-scale framework that integrates the flow models in different granularities.

A hierarchical structure is a natural structure for implementing the multi-scale

modeling. The underlying idea is to use a global flow model to capture the commonalities over a large domain, and this model is refined within each local domain to yield local models that describe the local characteristics more accurately. These local models can be further refined recursively if necessary, which will give rise to a hierarchy of models. Let $B_1, \ldots, B_m$ are local models that are generated by refining the model A, then A is called the *parent model* of $B_1, \ldots, B_m$; while $B_1, \ldots, B_m$ are called A's *children models.*

The construction of a multi-scale model hierarchy involves three stages:

1. *Domain subdivision.* Given a flow F_0 defined on a spatial domain D_0, when we intend to refine F_0, we first divide D_0 into several smaller regions $D_{1,1}, \ldots, D_{1,m}$ with $D_0 = \bigcup_{k=1}^{m} D_{1,k}$, on which the local models are based.

2. *Flow refinement.* After the local domains $D_{1,1}, \ldots, D_{1,m}$ are determined, we can generate m local flows $F_{1,1}, \ldots, F_{1,m}$ for these local domains, which can then be refined respectively.

3. *Flow Assembling.* Combine the refined local flows into a coherent global flow F_1. This process may involve the enforcement of some consistency constraints.

There are different methods for domain division and local flow assembling. In the following, I will discuss two methods that we may consider to adopt in our framework.

Disjoint division into polygonal cells

The first way is to divide each domain into polygonal sub-domains with a polygonal mesh. Different sub-domains do not overlap with each other except at the boundary, and each local flow is strictly confined within the given sub-domains. To ensure that the integrated flow is well-defined at the boundary, we need to enforce the *consistency constraints* at the boundary, which are in the following form

$$V_{F_a}(x) = V_{F_b}(x), \quad \forall x \in D_a \cap D_b. \tag{4.28}$$

Let $E_1, \ldots, E_L$ be the basis of the Lie algebra associated with the flow family, such that $V_{F_a} = \sum_{l=1}^{L} \alpha^l E_l$ and $V_{F_b} = \sum_{l=1}^{L} \beta^l E_l$, then the consistency constraints can be written in terms of flow coefficients as

$$\sum_{l=1}^{L} \alpha^l E_l(x) = \sum_{l=1}^{L} \beta^l E_l(x), \quad \forall x \in D_a \cap D_b. \tag{4.29}$$

We can see that they are linear constraints. Enforcing these consistency constraints results in a space of consistent flows, denoted by $\mathcal{CF}_1$, which is a subspace of $\mathcal{F}_1$, the joint space of local flows.

The dimensionality of the space of consistent flows in general depends on the graphical topology of the mesh, as well as the choice of the flow class.

Triangle mesh and consistent subspace

When a triangular mesh is employed (*i.e.* dividing the image plane into disjoint triangular cells) and the affine flows are used as local flows, we have more specific results.

Specifically, we partition the scene using a triangle mesh with m cells and n vertices, and attach each cell with an affine flow. Let $E_1, \ldots, E_L$ denote the basis of the associated Lie algebraic representation (base velocity fields). Then the local flow of the i-th cell can be represented by an L-dimensional Lie algebraic coefficient vector $\boldsymbol{\beta}_i = [\beta_i^1, \ldots, \beta_i^L]^T$.

Local flows may generate different velocities at shared vertices. Consider a vertex x shared by the i-th and j-th cells. In general, $V_{F_i}(x)$ may not equal $V_{F_j}(x)$, leading to discontinuities at a cell boundary.

To avoid such inconsistencies, we require that the local flows yield the same velocities at shared vertices, i.e. $V_{F_i}(x) = V_{F_j}(x)$, resulting in the *consistency constraints* of the local coefficients as

$$\sum_{l=1}^{L} (\beta_i^l - \beta_j^l) E_l(x) = 0. \tag{4.30}$$

If all triangles are non-degenerated, *i.e.* three vertices do lie on a line, then there are in total $(6m - 2n)$ independent consistency constraints, which give rise to a $2n$-

dimensional space of consistent flows.

The strategy of dividing domains into non-overlapping polygonal sub-domains is straightforward and simple to implement, however, it has two drawbacks:

1. The refined flow formed by stitching local affine flows is in general not smooth. In particular, $F_1(x,t)$ is not continuously differentiable at cell boundaries.

2. The space of consistent flows varies as the sub-domains change. This means that when a local modification is made to the triangle mesh, all consistency constraints need to be re-computed, making it difficult to apply it to the context with evolving domains.

Partition of unity

To overcome these difficulties, we also explore another method that is based on *partition of unity*, which is a concept originating from topology for constructing global continuous functions from locally defined functions. The basic idea is to combine local functions with overlapped smooth "window functions" that sum to unity.

Let $D_{1,1}, \ldots, D_{1,m}$ be open sub-domains that covers D_0, i.e. $D_0 = \bigcup_{k=1}^{m} D_{1,k}$, then the set of non-negative functions $\{w_k : D_0 \to \mathbb{R}\}$ is called a *partition of unity* subordinate to the cover if it satisfies

$$\sum_{k=1}^{m} w_k(x) = 1, \ \forall x \in D_0, \quad \text{and} \quad w_k(x) = 0, \ \forall x \notin D_{1,k}, \forall k = 1, \ldots, m. \tag{4.31}$$

Back to the multi-scale flow modeling problem. Let $F_{1,1} \ldots, F_{1,m}$ be the local flows defined respectively on the overlapping over-domains $D_{1,1}, \ldots, D_{1,m}$, then we can construct the complete flow F_1 over D_0 in terms of its infinitesimal generator as

$$V_{F_1}(x) = \sum_{k=1}^{m} w_k(x) \ V_{F_{1,k}}(x). \tag{4.32}$$

The flows constructed in this way are well-defined without the need to enforcing additional consistency constraints. In addition, if both $f_{1,k}$ and w_k are smooth, then

the derived flow must be smooth. Furthermore, we have the following property that makes it suitable in the context with evolving domains:

Proposition 4.1. *Let* $\{w_k : D_0 \to \mathbb{R}\}$ *be a partition of unity subordinate to a given cover* $D_{1,1}, \ldots, D_{1,m}$*, then if the domains are transformed by a diffeomorphism* T*, i.e.* $D_{1,k} \mapsto TD_{1,k}$*, then* $\{Tw_k : TD_{1,k} \to \mathbb{R}\}$ *is a partition of unity subordinate to the new cover, i.e. the sum-to-unity condition continues to hold, where* Tw_k *is defined by* $Tw_k(x) = w_k(T^{-1}x)$.

It means that domain evolution will not affect the validity of the flow if the weighting functions are transformed accordingly. In summary, using partition of unity method may incur more computation cost as multiple flows are involved in the overlapping area, which, however, brings forth a series of benefits, which, for example, include modeling flexibility, smoothness, and adaptivity to evolving environment where the domain of flow can change dynamically.

4.4 Stochastic Flow Model

To model real scenes, we are facing a complicated situation with a series of practical challenges. These challenges include noisy measurement, missing data, and errors arising in the feature extraction processes. To cope with these difficulties, we extended the algebraic formulation developed above to a stochastic model that allows objects to deviate from the ideal trajectories.

4.4.1 The Stochastic Flow Formulation

The stochastic model of a dynamic flow is formulated as a diffusion process characterized as follows:

$$x(t) = F(x_0, t) + \sigma B(t) \tag{4.33}$$

Here, $F(x_0, t)$ is a deterministic geometric flow as given by

$$\frac{dx(t)}{dt} = V_F(x(t)), \tag{4.34}$$

and $B(t)$ is the Brownian motion that accounts for the residual part due to deviation or measurement noise. This equation can be written equivalently, as

$$dx_t = V_F(x) + \sigma dB_t. \tag{4.35}$$

Let Δt be the time interval between two measurements of $x(t)$, then if Δt is sufficiently small, we have

$$x(t+\Delta t) - x(t) \simeq V_F(x) + \xi_{\Delta t}, \qquad \xi_{\Delta t} \sim \mathcal{N}(0, \sigma^2 \Delta t). \tag{4.36}$$

In practice, we will restrict F to a finite dimensional family, whose associated Lie algebraic space has basis $(E_1, \ldots, E_L)$, and express V_F as the linear combination of these bases. Then, we have

$$x(t+\Delta t) - x(t) \simeq \sum_{l=1}^{L} \alpha_l E_l(x) + \xi_{\Delta t}. \tag{4.37}$$

If x is observed, and a Gaussian prior is assumed for the Lie algebraic coefficients $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_L)$, as

$$\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_0). \tag{4.38}$$

Then, the maximum a posterior estimation of the flow, formulated as

$$\text{maximize} \quad \log p(\boldsymbol{\alpha}) + \sum_{i=1}^{n} \log p(x_i|\boldsymbol{\alpha}), \quad \text{w.r.t. } \boldsymbol{\alpha}, \tag{4.39}$$

will reduce to a regularized linear regression problem, as below

$$\text{minimize} \quad \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\alpha} + \frac{1}{2}(\sigma^2 \Delta t)^{-1} \sum_{i=0}^{n} \sum_{j=1}^{n_t} \left(x_i(t_j) - x_i(t_{j-1}) - \sum_{l=1}^{L} \alpha_l E_l(x_i(t_{j-1})) \right)^2. \tag{4.40}$$

Here, n_t is the number of time steps, and $t_j = j\Delta t$ is the j-th time step. This is a quadratic optimization problem and can be readily solved.

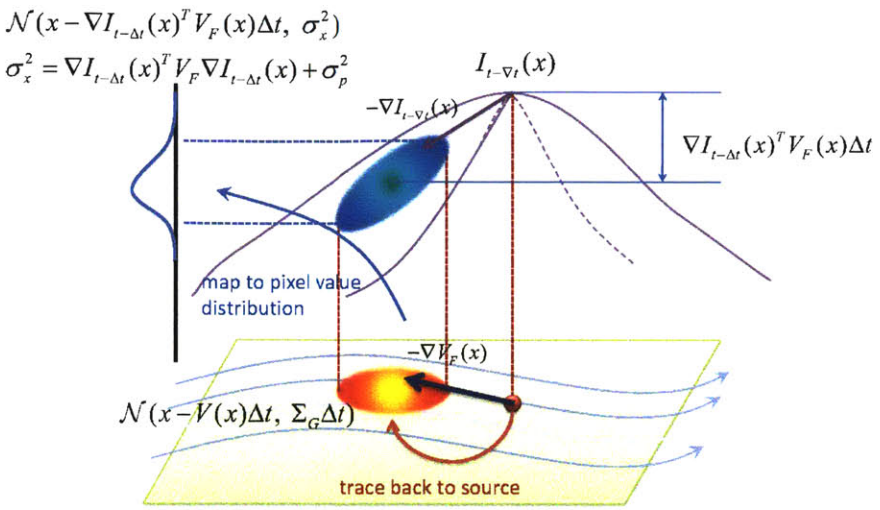


Figure 4-6: Each pixel in current frame is modeled as generated by moving a source pixel along the flow to current position. To get its distribution, we first trace the pixel backward along the flow to obtain the distribution of source point location, and then map it to the distribution of pixel values through the image. The additional term σ_p^2 is to capture the measurement noise of pixel values.

4.4.2 The Action of Stochastic Flow on Images

Now, we apply stochastic flows on images and develop a formulation for estimating flows from observed image sequences without the need of tracking individual points.

Let $I_{t_0}, \ldots, I_{t_J}$ be a sequence of image frames capturing a layer driven by a stochastic flow as in Eq.(4.35). As the stochastic flow is a Markov process, we have

$$p(I_{t_0}, \ldots, I_{t_J} | F, \sigma^2) = \prod_{j=1}^{J} p(I_{t_j} | I_{t_{j-1}}; F, \sigma_f^2). \tag{4.41}$$

Here, σ_f^2 is the variance coefficient of the Brownian motion. Assuming that observed pixels are independent conditioned on the previous frame, we get

$$p(I_{t_j} | I_{t_{j-1}}; F, \sigma_f^2) = \prod_{\mathbf{x} \in \mathcal{D}_I} p(I_{t_j}(\mathbf{x}) | I_{t_{j-1}}; F, \sigma_f^2). \tag{4.42}$$

Here $I_{t_j}(\mathbf{x})$ is the pixel value of I_{t_j} at location $\mathbf{x}$, and $\mathcal{D}_I$ is the set of all observed

pixel locations in the flow domain. Through back tracing (see Figure 4-6), we obtain

$$p(I_{t_j}(\mathbf{x})|I_{t_{j-1}}; F, \sigma_f^2) = \mathcal{N}(I_{t_{j-1}}(\mathbf{x}) - \mu_{\mathbf{x}},\ \sigma_{\mathbf{x}}^2). \tag{4.43}$$

with

$$\mu_{\mathbf{x}} = \nabla I_{t_{j-1}}(\mathbf{x})^T V_F(\mathbf{x})\Delta t, \tag{4.44}$$

$$\sigma_{\mathbf{x}}^2 = \sigma_f^2 \nabla I_{t_{j-1}}(\mathbf{x})^T \nabla I_{t_{j-1}}(\mathbf{x})\Delta t. \tag{4.45}$$

Here, we use a locally linear approximation as below in deriving the normal distribution above:

$$I(\mathbf{x} + \Delta\mathbf{x}) \simeq I(\mathbf{x}) + \nabla_{\mathbf{x}}\Delta\mathbf{x}. \tag{4.46}$$

Note here that Eq.(4.45) suppresses the influence of the pixels with high contrast neighborhood. With Lie algebraic representation, we can further expand each factor in Eq.(4.42) as

$$p(I_{t_j}(\mathbf{x})|I_{t_{j-1}}; F, \sigma_f^2) = \mathcal{N}\left(I_{t_j}(\mathbf{x}) - I_{t_{j-1}}(\mathbf{x}) \middle| -\sum_{l=1}^{L} \alpha^l \mu_{\mathbf{x}}^{(l)},\ \sigma_{\mathbf{x}}^2\right). \tag{4.47}$$

Here, $\mu_{\mathbf{x}}^{(l)} = \nabla I_{t_{j-1}}(\mathbf{x})^T E_l(\mathbf{x})\Delta t$. Therefore, we can model the changes of image pixels using pixel-wise normal distributions. At a particular pixel, the mean of its corresponding normal distribution depends on the flow coefficients, while the variance depends on the image gradient at the same location.

Again, the estimation of flow coefficients from a sequence of images can be formulated as an MAP problem, as follows

$$\text{maximize} \quad \log p(\boldsymbol{\alpha}) + \log \sum_{j=1}^{n_t} \log p(I_{t_j}|I_{t_{j-1}}; \boldsymbol{\alpha}). \tag{4.48}$$

Here, we exploit the fact a sequence of images generated according to a given stochastic flow model constitutes a Markov chain. When $p(\boldsymbol{\alpha})$ is a Gaussian prior, this problem reduces to a quadratic programming problem, which we can readily solve.

4.4.3 Integration of Observations

From the discussion above, it can be seen that under a stochastic flow formulation, the likelihood has a similar form when different types of observations are utilized (*e.g.* tracks of individual objects and image frames), in spite of their different generation processes. Specifically, we have

1. When point tracks are used, the likelihood of each track $\mathbf{t}$ can be written as

$$p(\mathbf{t}|\boldsymbol{\alpha}) = \prod_{j=1}^{n_t} \mathcal{N}\left(\mathbf{t}(j) - \mathbf{t}(j-1) \middle| \sum_{l=1}^{L} \alpha_l E_l(\mathbf{t}(j)),\ \sigma_B^2 \Delta t\right). \tag{4.49}$$

2. When image frames are directly worked on, the likelihood of each image frame, conditioned on the previous one, can be written as

$$p(I_j|I_{j-1}, \boldsymbol{\alpha}) = \prod_{\mathbf{x}\in\mathcal{D}} \mathcal{N}\left(I_j(\mathbf{x}) - I_{j-1}(\mathbf{x}) \middle| \sum_{l=1}^{L} \alpha_l \mu_{\mathbf{x}}^{(l)},\ \sigma_{\mathbf{x}}^2\right). \tag{4.50}$$

   Here, $\mu_{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$ are respectively the model-predicted mean and variance of the pixel change at $\mathbf{x}$.

For both cases, the likelihood terms can be written uniformly as products of $\mathcal{N}(\mathbf{y}_i|\mathbf{E}_i\boldsymbol{\alpha}, \boldsymbol{\Sigma}_i)$. Here, vy_i denotes the observed change, $\mathbf{E}_i$ denotes the matrix where each column represents the "base change" induced by a particular base flow. Thus, $\mathbf{E}_i\boldsymbol{\alpha}$ is the model-predicted change, as a linear combination of the base changes. Here, we call the triplet $(\mathbf{y}_i, \mathbf{E}_i, \boldsymbol{\Sigma}_i)$ an *observation entry.*

With the generic form of likelihood terms discussed above, one can integrate different types of observations whenever they are available from the scene. This would lead to a unified model with multiple Gaussian likelihood factors. As a result, flow estimation based on this model can be casted as a joint Gaussian inference problem.

4.4.4 Gaussian Process Prior over Complex Flows

Flows in natural scenes exhibit complex yet spatially coherent variations. Effective modeling of such flows requires a fine-grained mesh that may compromise the spatial

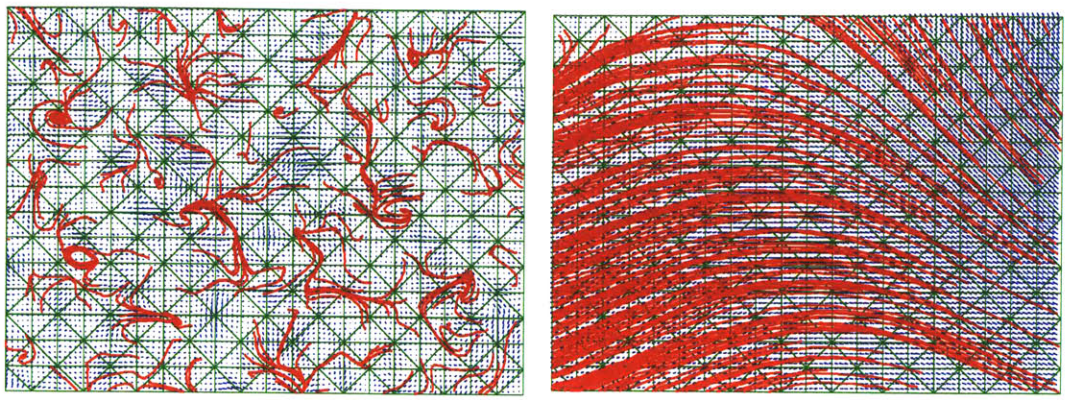


Figure 4-7: Here show the flows sampled from two prior models. The left one is sampled from the GP-prior with $\sigma_{GP} = 0$. In this case, essentially no spatial coherence is enforced. The right one is sampled from the GP-prior with $\sigma_{gp} = 300$ pixels. For each sample, 400 trajectories are simulated and shown as red curves.

coherence. Consequently, we incorporate a Gaussian process (GP) [80] as a prior on flows to enforce long-range spatial coherence while retaining modeling flexibility.

Consider a flow constructed as a combination of m local flows, each covering a cell, *i.e.* a region of the image plane. Let $\mathbf{c}_i$ be the circumcenter of the i-th cell, and $\boldsymbol{\beta}_i = [\beta_i^1, \ldots, \beta_i^L]^T$ be the associated local Lie algebraic representation. The covariance function is defined over the local representations, such that

$$\mathrm{cov}(\beta_i^l, \beta_j^l) = \sigma_\beta^2 \exp\left(-\frac{1}{2}\frac{||\mathbf{c}_i - \mathbf{c}_j||^2}{\sigma_{gp}^2}\right). \tag{4.51}$$

This leads to a Gaussian prior $\mathcal{N}(\mathbf{0}, \mathbf{G}_\beta)$ of the concatenated local representation, in which $\mathbf{G}_\beta$ is an $mL \times mL$ matrix. Recall that under consistency constraints, we can write $\boldsymbol{\beta} = \mathbf{U}\boldsymbol{\alpha}$ with a d_C-dimensional coefficient vector $\boldsymbol{\alpha}$. Hence, the GP-prior of a consistently stitched flow can be further derived as $p(\boldsymbol{\alpha}) = \mathcal{N}\left(\mathbf{0}, (\mathbf{U}^T\mathbf{G}_\beta^{-1}\mathbf{U})^{-1}\right)$. Figure 4-7 compares the effects of GP-prior illustrating its crucial role in preserving spatial coherence when a fine-grained mesh is used.

4.4.5 Multiple Concurrent Flows

Multiple coexisting flows are common in natural scenes. To characterize motion in such scenes, we consider an extended model that comprises M flows, including a

"background flow" that covers the entire image plane.

To estimate the multi-flow model from observed scenes, we have to associate each observation to a particular flow. To this end, we introduce a latent variable z_i for each observation entry to indicate which flow it is generated from.

In practical applications, observations are not independent. Those that are spatially close to each other are more likely to come from the same geometric flow. Taking this into account, we further incorporate an MRF among z_i to encourage assignment of the same label to observations near each other. Specifically, this MRF is formulated as follows

$$p(Z|\text{MRF}) = \frac{1}{C_{MRF}} \exp\left(- \sum_{(i,j):i\sim j} w_{ij} \mathbb{I}(z_i \neq z_j) \right). \tag{4.52}$$

Here, $i \sim j$ means i and j are spatial neighbors of each other, w_{ij} is an affinity value that reflects how close they are, and C_{MRF} is a normalization constant.

Moreover, erroneous observation entries, such as wrong tracks due to mis-association across frames and image gradients computed across sharp boundaries, are sometimes used, which may severely bias the estimation of flow coefficients. Therefore, we introduce a binary variable g_i for each observation entry. $g_i = 1$ indicates that the i-th observation entry is an inlier. The use of inlier indicators may help to suppress the influence of erroneous observations, thus improving the model's robustness.

Combining the observation models, flow prior, and the MRF of flow assignments, we obtain a joint probabilistic model as follows. Suppose there are M independent flows and N observation entries. The joint probabilistic formulation is then given as below

$$p(Z|\text{MRF}) \prod_{k=1}^{M} p(\boldsymbol{\alpha}_k|\text{GP}) \prod_{i=1}^{N} p(g_i|c_i) p(\mathbf{y}_i|\mathbf{E}_i, z_i, g_i; \mathcal{F}). \tag{4.53}$$

Here, $\mathcal{F}$ denotes the flow model, which comprises the Lie algebraic coefficients of the flows. The likelihood term is given by

$$p(\mathbf{y}_i|\mathbf{E}_i, z_i, g_i; \mathcal{F}) = \begin{cases} \mathcal{N}(\mathbf{y}_i|\mathbf{E}_i\boldsymbol{\alpha}_{z_i}, \boldsymbol{\Sigma}_i) & (g_i = 1), \\ Q & (g_i = 0). \end{cases} \tag{4.54}$$

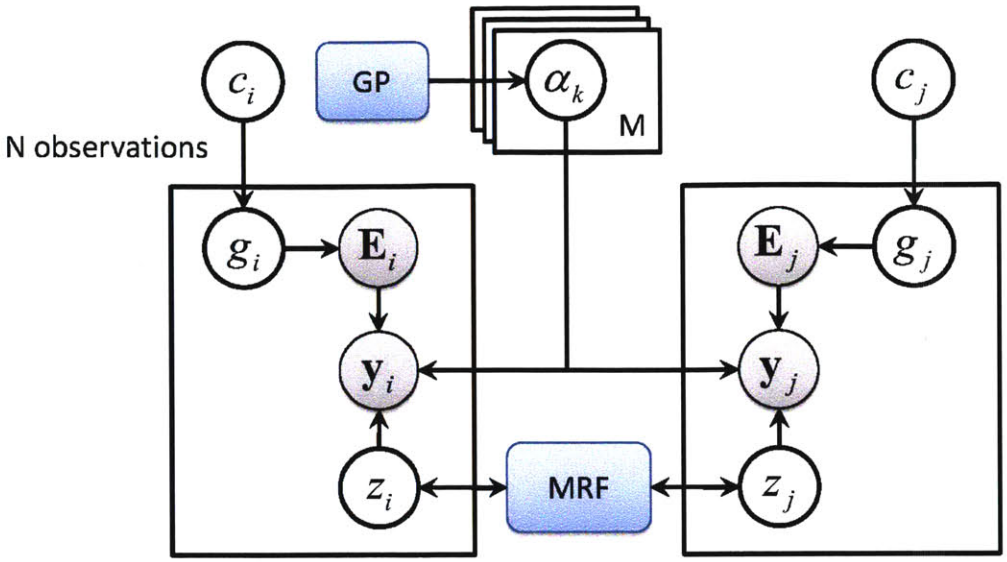


Figure 4-8: This graphical model incorporates the generative observation model and the GP-prior of the flows. Here, each observation entry is associated with a label variable z_i that indicates which flow it is generated from and a binary variable g_i that indicates whether it is a valid observation. The label variables are connected to each other through an MRF, while the distribution of g_i is independent, characterized by a prior confidence c_i, *i.e.* the prior probability of $g_i = 1$.

The graphical model of the joint probabilistic framework is shown in Figure 4-8.

The estimation can be done using variational EM based on a mean-field approximation[1] of the posteriori. The algorithm iteratively re-estimates the flow coefficients using the relabeled observation entries, re-assigns each observation to the updated models, and updates the inlier probabilities. Graphcut [14] is used for re-labeling in the variational E-steps.

4.5 Experiments

In contrast to much of the prior work on motion analysis where accurate estimation of local velocities is the major concern, this work aims at discovering the persistent motion patterns that drive the evolution of a scene. Hence, the capability of being generalized to model unseen observations is an important aspect to examine.

[1]Section 2.3 provides a detailed exposition of the mean-field approximation and variational EM algorithm.

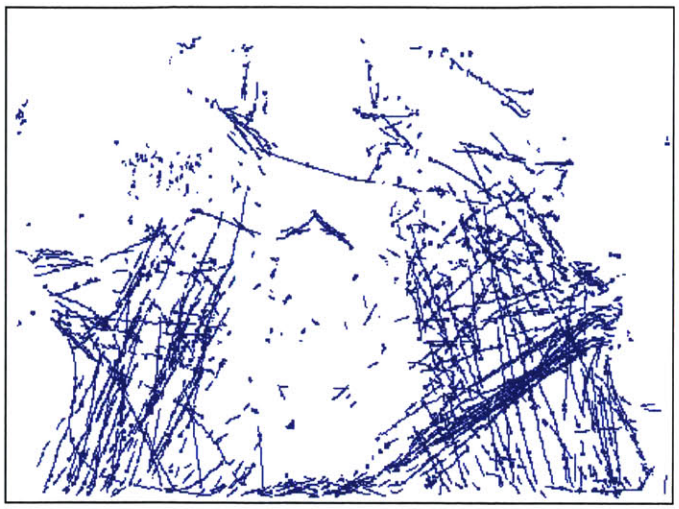


Figure 4-9: The plot of all extracted local motions from the New York Grand Central station.

Figure 4-10: Three are three representative flows discovered by the Lie algebra based flow model. The region that is not covered by the flow is masked. The blue arrows indicate the flow field, and a subset of persons governed by the flow is highlighted with red boxes.

4.5.1 Analyzing Crowd Motion Patterns

The experiments are performed on a video captured in New York's Grand Central station. The video sequence is 15 minutes duration captured at 24 fps at an image resolution of 1440×1080. The first 1000 frames are used to initialize the model, and the 18000 frames that follow are processed using the tracking algorithm of [93]. In such scenes, one expects a degree of regularity of motion due to a variety of factors including the movement of large crowds of people negotiated in a confined space or common entrances and exits. Our aim is to capture the aggregate motion patterns solely from local motion observations.

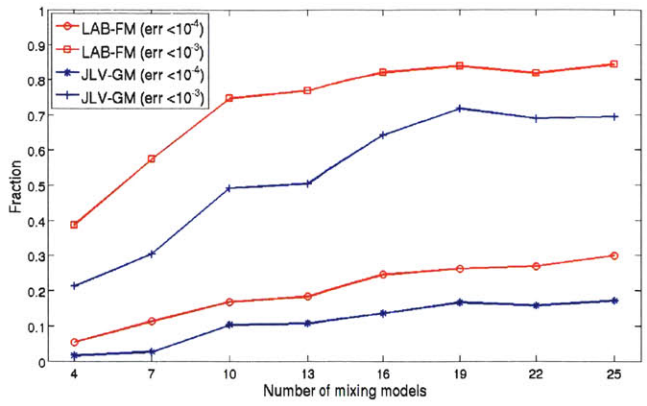


Figure 4-11: This figure compares the motion prediction performance of LAB-FM and JLV-GM on testing samples. The x-axis here is the number of mixture components, and the y-axis quantity here is the fraction of observations within given errors from the model prediction.

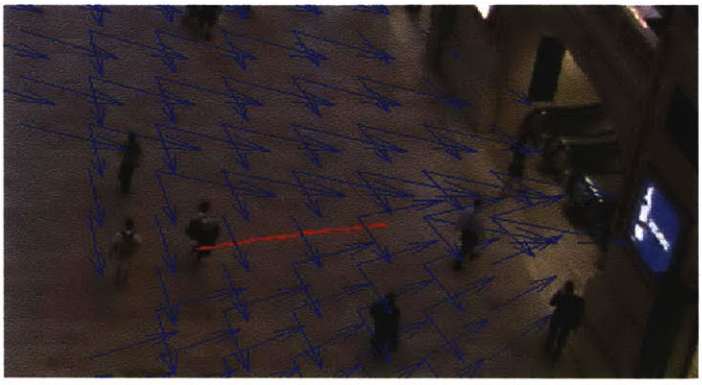


Figure 4-12: An example of outlier detected by the flow model. Here, the trajectory of the person (highlighted with red color) is clearly different from what the flow model may predict at his location.

In such scenes, tracking errors commonly occur due to a variety of issues such as occlusions and individuals crossing paths. In Figure 4-9 we show roughly 10% of the extracted local motions. It can be observed that, even in the presence of errors, there is observable structure in the motions. Additionally, one can observe that structured motion fields overlap with each other.

We apply our flow model in order to recover these flows, setting the number of flows to $M = 16$. The noise covariance matrices are all set to $\sigma^2\mathbf{I}$ with $\sigma = 3$. This setting is based on rough estimation, and the final performance is relatively insensitive to the value of σ here. Figure 4-10 illustrates several of the learned flows, where affine

deformations are represented as flow fields (blue). Individual motions associated with this flow (red) demonstrate that the affine model is able to capture aggregate motion over a large region, despite the fact that individuals following these patterns appear distinct locations and times and walk along different paths to different locations.

We compare our results to a baseline method, which groups the locations and local motions based on their proximity and models each groups with a prototypical motion. For conciseness, we refer to our method as "LAB-FM" (Lie Algebra Based Flow Model) and the comparison model as "JLV-GM" (Joint Location-Velocity Group Model). In order to have a fair comparison, JLV-GM is formulated similarly, so as to cope with noise and outliers. The consequence being that the essential difference arises from exploiting the group structure in the Lie Algebra space.

While mean squared error is a widely used metric to measure prediction accuracy, the effect of small amount of outliers can dominate the performance measurement. To mitigate such influence, we, instead, measure the performance in terms of the fraction of samples whose squared errors are below some given threshold. Setting the thresholds to 10^{-4} and 10^{-3}, and the number of mixing models M to different values, we obtain the performance curve shown in figure 4-11. The results clearly show that the performance increases as we add more components, and our LAB-FM consistently outperforms JLV-GM, that is, at a given threshold, the fraction of testing samples which are below this error threshold is higher for the LAB-FM than for JLV-GM. When $M = 16$, using our model, 24.6% and 82.2% of all testing samples are with squared errors lower than 10^{-4} and 10^{-3}, while the fractions for JLV-GM are only 13.6% and 64.2%.

While outliers may be indicative of many things, they primarily correspond to motions which differ from the typical behavior of individuals in the scene. Figure 4-12 shows one of the outliers. There are three dominant flow fields in the scene. The "outlier", however, walks towards the escalator (a converging destination for two of the flows from either top or bottom of the scene) from a horizontal direction. The implication is that during the observation period, the majority of individuals in this region either enter the escalator from one of two directions or pass it by. Additionally,

the analysis indicates that during this period, very few people entered the scene from the escalator or from the bottom of the scene.

4.5.2 Modeling Flows in General Dynamic Scenes

Next, we test the geometric flows on more complex scenes, where each flow is composed of multiple local flows. In particular, we conducted experiments on videos from DynTex database [4] and UCF Crowd Analysis database [2]. These videos represent a wide spectrum of dynamic scenes.

Experiment Settings

We use consistently stitched flows to model the motion patterns in each video. Each flow is established on a triangle mesh that covers the entire scene and parameterized by the Lie algebraic representation. To generate the triangle mesh, we first make a rectangle mesh with 5 rows and 6 columns, and divide each rectangle cell into two triangles.

Affine flow family is chosen to be the basic flow family for describing the motion within each cell. This family is associated with a 6-dimensional Lie algebra. While we found that this setting suffices to obtain good results on the testing videos, generally there is no restriction to the choice of basic flow family and mesh topology. One can also use other flow families with our framework to model more complex motion. The representation dimension (the dimension of the consistent subspace) is $L = 84$, which is much smaller than that of the dense velocity map. In addition, we use a GP prior to enforce long-range spatial coherence, where σ_{gp}, which controls the range of correlation, is set to 100.

We use multiple concurrent flows to model a dynamic scene, including one that corresponds to the static background. The number of flows in each scene is given. To initialize the algorithm, we manually specify a "seed area" for each flow, and the observations in these areas are used to estimate the initial flow models. Besides, we set the prior confidence of inlier to 0.9 for each observation entry. Flow segmentation,

inlier probabilities, and flow models are then iteratively updated until convergence.

To compare our approach with traditional local motion estimation methods, we implement an optical flow estimation algorithm with multi-scale search and local smoothness. Moreover, we adapt the algorithm to incorporate multiple frames in estimation for fair comparison. This is accomplished by assuming a time-invariant velocity v at each location, and integrating the term $||\frac{\partial I}{\partial t} + v^T \nabla I||^2$ for every pair of consecutive frames into the objective function. The design parameters of the optical flow, including the local search range and the coefficients in the smoothness terms, are optimized by cross validation.

Collective Behavior of Moving Objects

The algorithms are first evaluated on the scenes that comprise groups of moving objects, such as people, and vehicles. To test the generalization performance, for each video, we first manually trace 20 trajectories, whose nominal duration is 100 frames, as ground truth, and then use the first 20 frames to estimate the motion models. These models are then used to simulate the trajectories starting from the initial positions as those in the manually traced ones. The performance is measured by the deviation of the simulated trajectories from the ground truth.

Figure 4-13 and Figure 4-14 shows the results. The results shown in the first row are obtained on a scene with cars moving along a high way. We see that the optical flow is over-fitted to the short-time behavior of individual cars: (1) it only extracts motion in the places where cars are passing by during first 20 frames; (2) the velocity map lacks spatial coherence. For the same example, the geometric flow accurately captures the collective behavior of the cars, while preserving spatial coherence. Note that the subtle variation of the moving direction of the cars is precisely captured in the flow model.

We also evaluate the trajectory prediction performance, observing that the predicted trajectories simulated on the optical flow field quickly deviate from the ground truth; while the ones yielded by geometric flow are much more accurate. The plotted error curves indicate that the average deviation due to the optical flow is more than

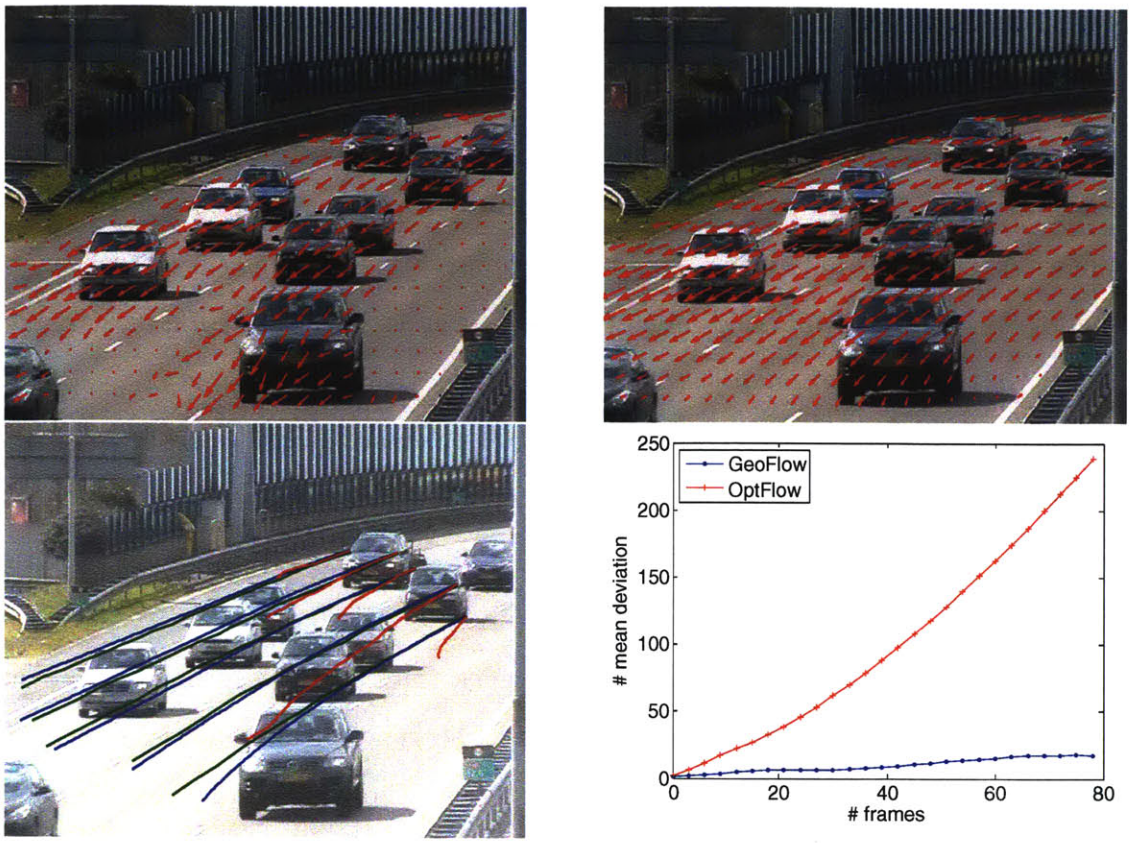


Figure 4-13: This figure shows the motion analysis results obtained from a scene with cars moving on a road. The first row shows the results respectively obtained using optical flow (left) and geometric flow (right), which are visualized in form of velocity fields. The left picture of the second row shows a subset of predicted trajectories (the blue curves are yielded by geometric flows, the red ones are yielded by optical flows, while the green ones are ground-truth derived by manual labeling. The fourth picture compares the trajectory-prediction error quantitatively.

8 times larger than that due to the geometric flow.

The second row shows the scene with a crowded group of athletes running along a circular path. Similar observations are obtained in this example. Again, due to its local focus, the motion field produced by optical flow lacks spatial coherence and doesn't generalize well, while geometric flow yields much better generalization performance.

Continuous Motion Patterns

The tests are also done on modeling continuous motion patterns, such as flowing water and deforming objects.

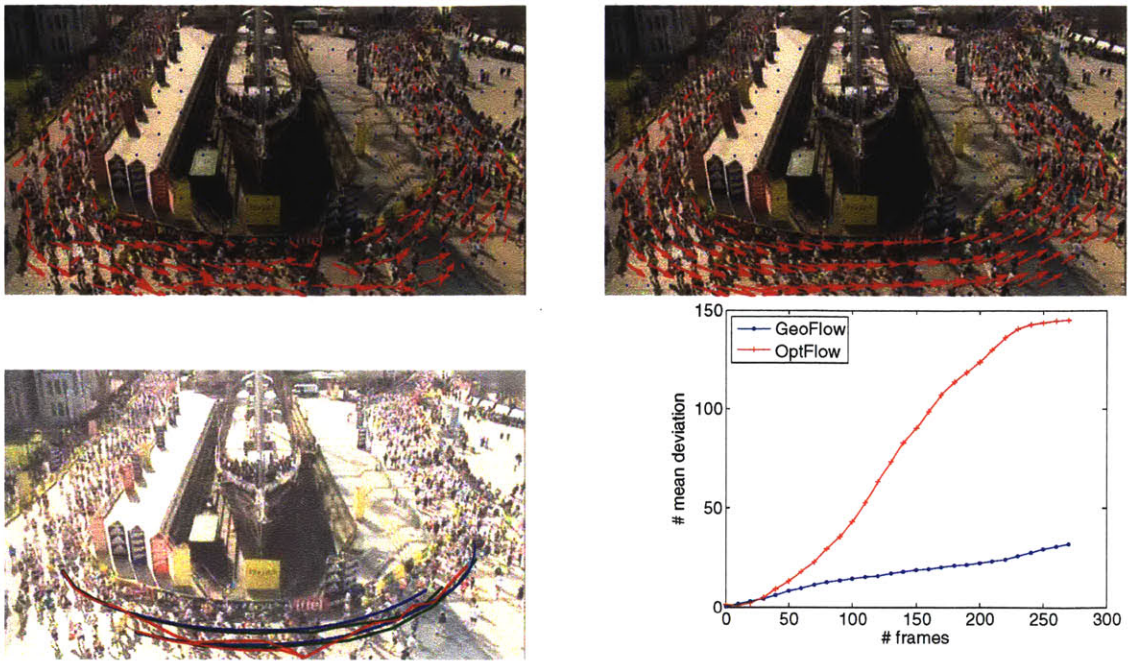


Figure 4-14: This figure shows the motion analysis results obtained from a scene with a large group of athletes running on tracks. The first row shows the results respectively obtained using optical flow (left) and geometric flow (right), which are visualized in form of velocity fields. The left picture of the second row shows a subset of predicted trajectories (the blue curves are yielded by geometric flows, the red ones are yielded by optical flows, while the green ones are ground-truth derived by manual labeling. The fourth picture compares the trajectory-prediction error quantitatively.

In Figure 4-15, the first column shows a mountain spring comprised of several sections with different motion patterns. To model this spring, we use four concurrent flows. The second column shows a disc rotating in a very high speed, whose appearance is severely blurred. The water transparency and the blurred texture on the disc lead to great challenges for motion estimation. In the face of such difficulties, optical flow performs poorly, resulting in meaningless motion patterns. Nonetheless, the geometric flow still works well. The subtle variation of the water flowing direction is precisely modeled while the spatial coherence is well preserved. The rotation of the disc in the right column is also successfully captured by the geometric flow.

As there are no discrete targets that can be tracked in these scenes, we use frame prediction to measure the performance. Concretely, we generate the frames from their preceding frames based on the predicted motion, which are then compared with the actual frames, in terms of average pixel-wise frame prediction error. The

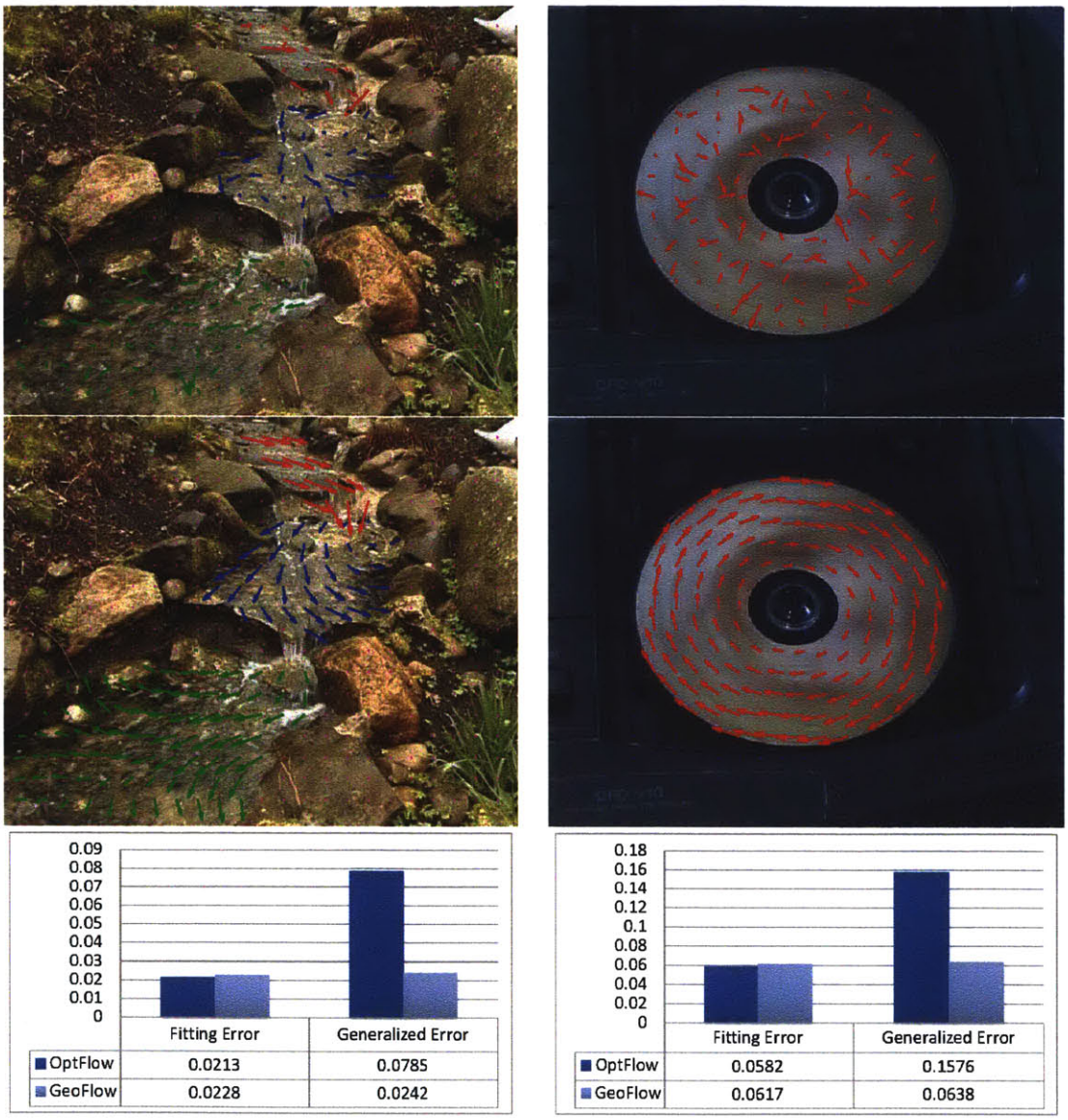


Figure 4-15: In this figure, the first column shows the results obtained on modeling the flowing water in a mountain spring. The second column shows that on a rotating disc. The bottom row shows two charts, giving the average fitting errors and generalization errors obtained from the corresponding example.

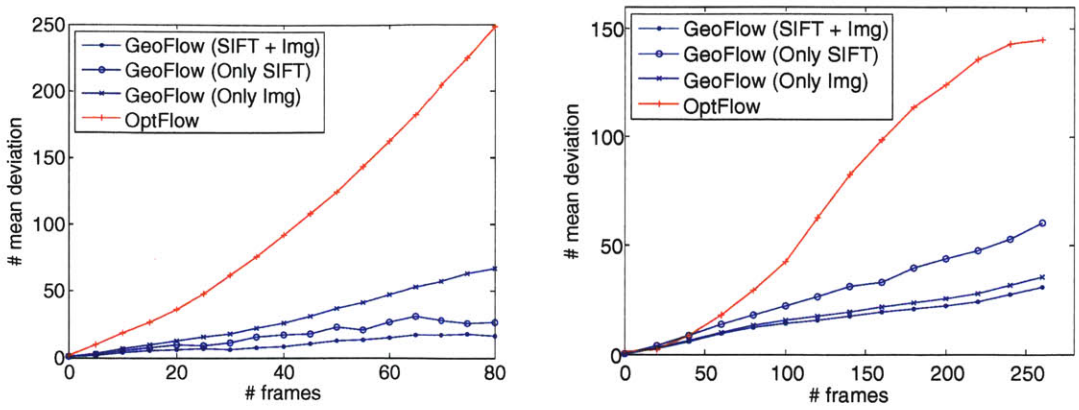


Figure 4-16: The trajectory prediction errors with different types of observations. The left and right charts are respectively obtained from the scene with moving cars and that with running athletes.

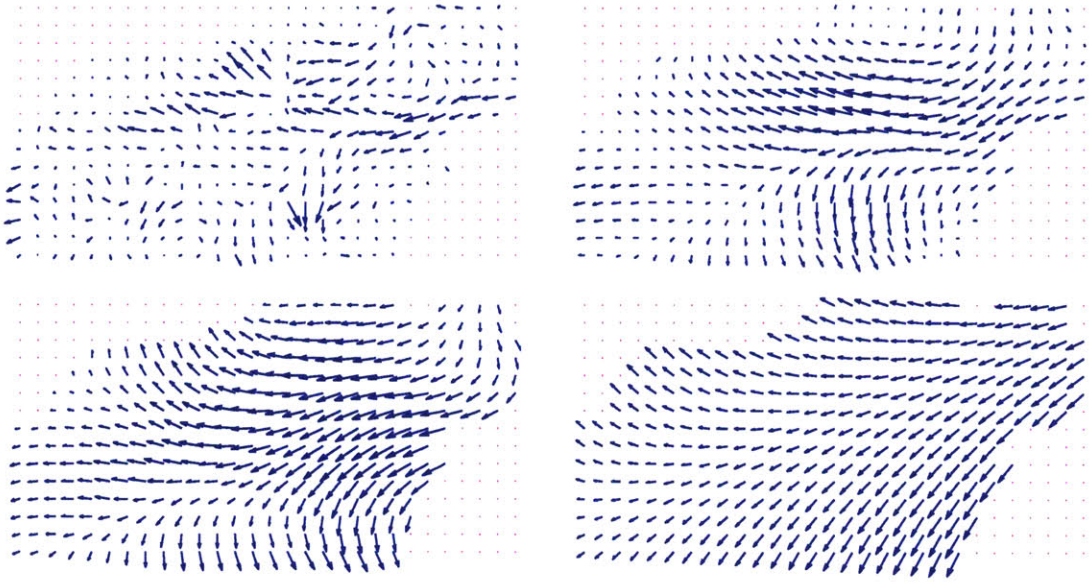


Figure 4-17: The figure shows the motion patterns of the bottom-right part of the mountain spring estimated under different settings. From left to right, the results are obtained by optical flow, geometric flows with σ_{gp} set to 0, 100, 10000 respectively.

performance is measured respectively for training frames and testing frames, which respectively reflect the fitting accuracy and generalization performance. We see that while geometric flows fit slightly less accurately to the training frames, they generalize remarkably better than the optical flow.

Study of Modeling Choices

To study how the use of different observations affects the performance, we test three settings: (1) only using image frames, (2) only using SIFT pairs, and (3) using both. Figure 4-16 compares the trajectory prediction errors under these settings, as well as that due to optical flow estimation.

The results show that these two types of observations are complementary, which are respectively suitable for different scenes (or different regions of a scene). Generally, point pairs perform better in the scenes with structured appearance where they can be accurately located; while pixel differences are more reliable for the scenes with smooth textures. However, in either case, the combination of both leads to further improvement, which reflects their complementary nature.

We also studied the influence of Gaussian process prior. In particular, we test the framework with different values of σ_{gp} to study how GP-prior influences the estimation. In Figure 4-17, we see that the motion pattern becomes more coherent as σ_{gp} increases. When σ_{gp} approaches infinity, local flow of every cell is restricted to be the same, resulting in a uniformly affine field. When σ_{gp} approaches zero, long-range consistency is no longer enforced. While the result obtained under this setting is less coherent than that with GP utilized, it still preserves the coherence within each cell and the consistency between neighboring cells, and thus is better than that of the optical flow.

4.6 Summary

In this chapter, we introduced a new model to characterize persistent motion patterns using geometric flows, and derived a vector space representation of flows based on

Lie algebraic analysis. This representation greatly simplifies the use of probabilistic modeling and inference on flows. We also developed a stochastic flow model and extended it through domain sub-division.

We conducted experiments to test this new approach on real videos, and compare it with optical flow estimation. The results clearly demonstrate that geometric flows, formulated with the goal of capturing coherent patterns over both space and time, perform substantially better than optical flows, and thus they are more effective in persistent motion modeling.

Chapter 5

Dynamic Bayesian Nonparametrics

In the vision models developed in this thesis, among many others, mixture models are used to capture complex distributions. For example, the appearance model presented in Chapter 3 uses it to approximate the patch manifold, while the motion model presented in Chapter 4 uses it to describe scenes with multiple flows. Classic formulations, such as finite mixture models, typically require the number of components to be specified prior to model estimation, leading to difficulties in many practical applications where such number is unknown or difficult to estimate a priori.

An important family of methodologies to address this issue – *Bayesian nonparametrics* – has been developed in past decades and becomes increasingly popular recently. As opposite to parametric models, nonparametric models do not assume a fixed structure in the formulation, thus allowing the model to grow in size to accommodate the need to characterize data of varying complexities.

With an aim to provide more flexible models to describe complex visual phenomena, we consider Bayesian nonparametric methods in this Chapter. We first review existing Bayesian nonparametric methods, particularly focusing on the ones based on Dirichlet processes (DPs). Then, we propose a new construction to overcome several difficulties that current constructions may encounter under a dynamic setting, which leads to a Markov chain of DPs that allows dynamic changes in a variety of ways.

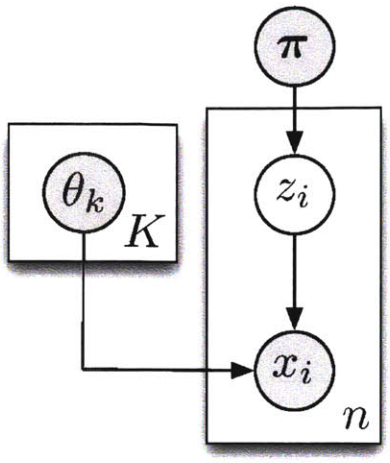


Figure 5-1: The graphical representation of a finite mixture model. The mixture comprises K component models, respectively with parameters $\theta_1, \ldots, \theta_K$. Data samples are generated independently from this model. In particular, to generate the i-th sample, z_i is first drawn from $\boldsymbol{\pi}$, and then the corresponding component $\mathcal{G}(\theta_{z_i})$ is used to generate x_i.

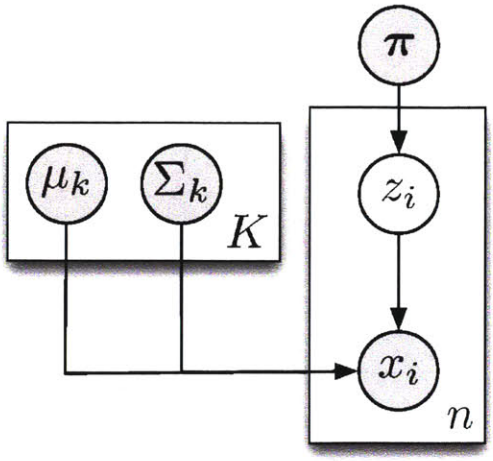


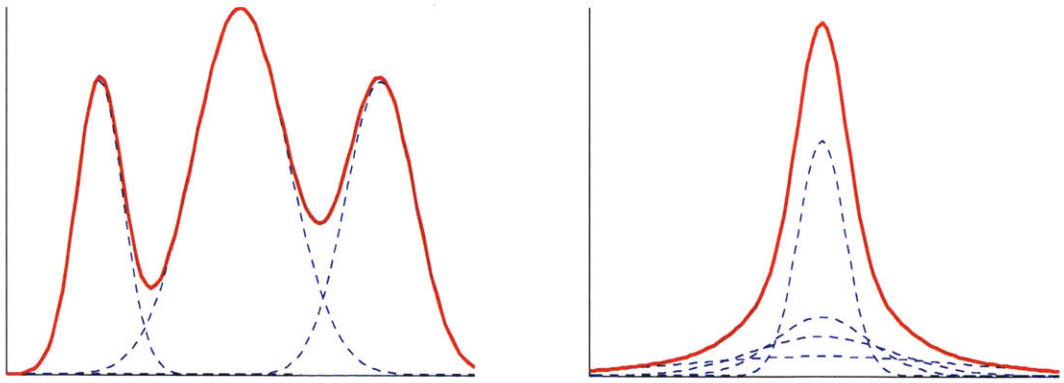
Figure 5-2: The graphical representation of a Gaussian mixture model, which consists of K Gaussian components. Each Gaussian component (say the k-th one) is characterized by a mean vector μ_k and a covariance matrix Σ_k. With this model, each data point is drawn independently from a particular Gaussian distribution, chosen from a discrete distribution $\boldsymbol{\pi}$.

5.1 Finite Mixture Models

In statistics, a *mixture model* is a probabilistic model composed of multiple simpler models (called *components*) to represent complex data distributions. Mixture models are very effective in modeling data with the presence of sub-populations that exhibit different statistical characteristics. In computer vision, mixture models have been widely used for different tasks, such as object recognition [63], scene categorization [16], dynamic tracking [94], shape representation [21], image segmentation [64], and activity recognition [69].

5.1.1 Generic Formulation

Consider a set of data points $x_1, \ldots, x_n$. A classic formulation of mixture model assumes that the data are independently generated from K parametric component



(a) Multi-modal distribution (b) Heavy-tail distribution

Figure 5-3: This figure shows two examples of using Gaussian mixtures to approximate other distributions: **(a)** a distribution with three modes is approximated by a mixture model comprised of three Gaussian components. **(b)** a heavy-tailed distribution is approximated by a mixture of four Gaussian components with zero mean and different variances (this is also called a *Gaussian scale mixture*).

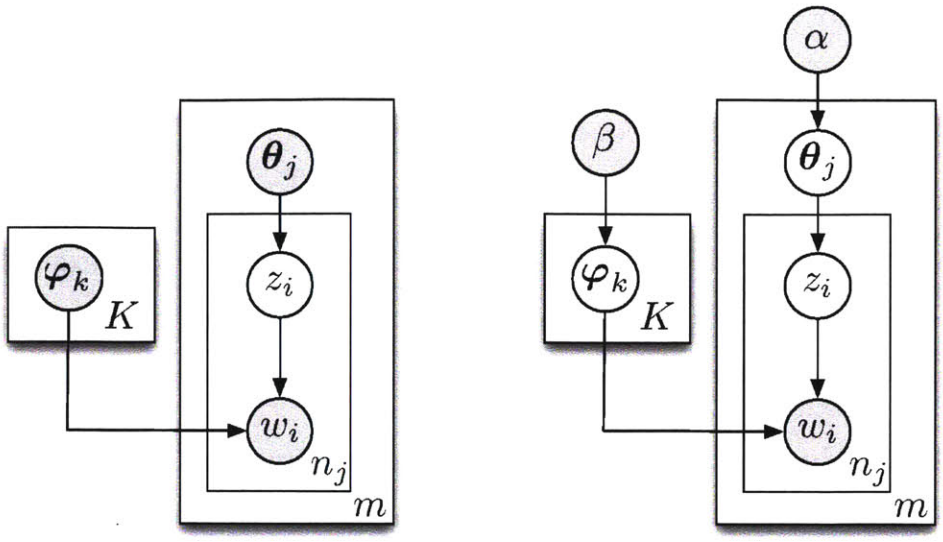
models, respectively with parameters $\theta_1, \ldots, \theta_K$.

$$z_i \sim (\pi_1, \ldots, \pi_K), \quad x_i \sim \mathcal{G}(\theta_{z_i}), \quad \text{for } i = 1, \ldots, n. \tag{5.1}$$

Figure 5-1 shows the graphical model of this generative process. Here, to generate a sample x_i, a particular component (indicated by z_i) is first chosen from a discrete distribution, the corresponding parametric model, denoted by $\mathcal{G}(\theta_{z_i})$ is then used to produce the sample. Hence, the probability density function of the mixture distribution can be written as a convex combination of the component pdfs, as

$$p_M(x|\boldsymbol{\pi}, \Theta) = \sum_{k=1}^{K} \pi_k f(x; \theta_k). \tag{5.2}$$

Here, $f(x; \theta_k)$ is the pdf of the k-th component. The model introduced above is called a *finite mixture model*, which requires K, the number of mixture components to be fixed prior to model estimation.



Probabilistic Latent Semantic Indexing (PLSI) Latent Dirichlet Allocation (LDA)

Figure 5-4: This figures show the graphical representation of two topic models under different formulations: **(a)** is probabilistic latent semantic indexing, where each document is associated with a document-specific mixture of topics $\boldsymbol{\theta}_j$. **(b)** is latent Dirichlet allocation, which extends PLSI by introducing a Dirichlet prior over the topic distributions. In addition to this, a Dirichlet prior is often incorporated as the prior of the word distributions.

5.1.2 Specific Examples: GMM and Topic Models

Generally, components in a mixture model can be arbitrary distributions. Two most common choices are *Gaussian distribution* and *Multinomial distribution*, which respectively lead to a *Gaussian mixture model* and a *Topic model*.

1. **Gaussian Mixture Model.** When the component models are Gaussians, a finite mixture model is often called a *Gaussian mixture model (GMM)*. As shown in Figure 5-2, each sample from a Gaussian mixture model can be generated as follows:

$$z_i \sim (\pi_1, \ldots, \pi_K), \quad x_i \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i}). \tag{5.3}$$

Here, μ_k and Σ_k are respectively the *mean vector* and the *covariance matrix* of the k-th Gaussian component. In practice, for the purpose of computational efficiency or for increasing the reliability of the estimated parameters, some restrictions are often imposed, *e.g.* all components share the same covariance

matrix, or the covariance matrices are diagonal matrices, etc. One can also regularize the estimation by placing a *normal inverse Wishart distribution* as the prior over the component parameters.

Mixtures of Gaussians are often used to approximate complex distributions arising in real world problems. Figure 5-3 shows two typical examples, where GMMs are respectively used to approximate a distribution with multiple modes and a heavy-tail one.

2. **Topic Model.** Topic models are also an important mixture model, which has been widely used in natural language processing, machine learning, and other fields. A *topic model* typically consists of multiple topics, each associated with a multinomial distribution over a vocabulary. This model considers a document as generated by a mixture of topics, and particularly, each word therein is independently drawn from one of the topics.

   There are various forms of topic models. A representative one, as shown in Figure 5-4(a), is *Probabilistic Latent Semantic Indexing (PLSI)* [46]. In this formulation, there is a set of K topics, each associated with a distribution over the vocabulary, respectively denoted $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_K$. In addition, each document D_j is characterized by a document-specific mixture of topics $\boldsymbol{\theta}_j$ that generates the words in the document, as

$$z_{ji} \sim \boldsymbol{\theta}_j, \quad w_{ji} \sim \boldsymbol{\varphi}_{z_{ji}}, \quad \forall i = 1, \ldots, n_j, \quad j = 1, \ldots, m. \tag{5.4}$$

   Blei *et al.* extends this model and develops *Latent Dirichlet Allocation (LDA)* [13], as shown in Figure 5-4(b). By introducing a Dirichlet prior over the document-specific topic distributions (*i.e.* $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m$), LDA provides a complete probabilistic interpretation of the generative process of documents.

   Whereas topic models were originally developed for document analysis, they were also used to model visual scenes [63], where each local feature in an image is considered as a visual word.

5.1.3 Estimation of Finite Mixture Models

Given a set of data $x_1, \ldots, x_n$, one can estimate the parameters of a finite mixture model using maximum likelihood estimation (MLE), that is, to solve the following optimization problem:

$$(\hat{\boldsymbol{\pi}}, \hat{\Theta}) = \underset{\boldsymbol{\pi}, \Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p_M(x_i | \boldsymbol{\pi}, \Theta). \tag{5.5}$$

Generally, there is no analytic solution to this problem. One can find the optimal solution using *expectation-maximization*, an iterative algorithm for finding maximum likelihood solutions for models with latent variables (see Chapter 2 for more details).

In a finite mixture model, each data sample x_i is associated with a latent variable $z_i \in \{1, \ldots, K\}$. Given the model parameters $(\boldsymbol{\pi}, \Theta)$ and the observation x_i, the posterior probabilities of z_i are given by

$$\Pr(z_i = k | x_i; \boldsymbol{\pi}, \Theta) = \frac{\pi_k f(x_i; \theta_k)}{\sum_{l=1}^{K} \pi_l f(x_i; \theta_l)}. \tag{5.6}$$

The E-M algorithm involves a K-dimensional vector $\mathbf{q}_i$ for each sample x_i to represent these posterior probabilities. Then the E-steps and M-steps are respectively given as follows.

1. **E-step.** Update the values of $\mathbf{q}_i$ for each $i = 1, \ldots, n$, as

$$\mathbf{q}_i^{(t)}(k) \leftarrow \frac{\pi_k^{(t-1)} f(x_i; \theta_k^{(t-1)})}{\sum_{l=1}^{K} \pi_l^{(t-1)} f(x_i; \theta_l^{(t-1)})}, \quad k = 1, \ldots, K. \tag{5.7}$$

2. **M-step.** Update the model parameters, including both $\boldsymbol{\pi}$ and Θ, as

$$\pi_k^{(t)} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \mathbf{q}_i(k), \quad k = 1, \ldots, K, \tag{5.8}$$

and

$$\theta_k^{(t)} \leftarrow \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{q}_i^{(t)}(k) f(x_i; \theta), \quad k = 1, \ldots, K. \tag{5.9}$$

The specific form of formulas to update component parameters $\theta_1, \ldots, \theta_K$ depends on the specific form of the component models. Take Gaussian mixture model for example, Eq.(5.9) reduces to the update of the mean vectors and the covariance matrices:

$$\mu_k^{(t)} \leftarrow \frac{1}{\pi_k^{(t)}} \sum_{i=1}^{n} \mathbf{q}_i^{(t)}(k) x_i \tag{5.10}$$

$$\Sigma_k^{(t)} \leftarrow \frac{1}{\pi_k^{(t)}} \sum_{i=1}^{n} \mathbf{q}_i^{(t)}(k)(x_i - \mu_k^{(t)})(x_i - \mu_k^{(t)})^T. \tag{5.11}$$

It is important to note that the E-M algorithm described above is a coordinate ascent algorithm that would converge to a local optima. A simple strategy that may lead to better solutions is to run the procedure many times with random initialization and choose the result that yields the highest joint likelihood.

5.2 Dirichlet Process Mixture Models

A major problem with finite mixture models is that they require the number of components to be specified before the models are estimated on data. This may lead to difficulties in the application where this number is unknown or difficult to estimate. To address this issue, nonparametric mixture models have been developed, notably the *Dirichlet Process Mixture Model*, also known as *DPMM* [73, 99]. As a DP mixture model may contain infinitely many components, it is also called an *infinite mixture model* in some early work [79].

5.2.1 Dirichlet Processes

First of all, we briefly review the concept of Dirichlet process (DP) and how a mixture model can be constructed based on a DP. Formally, a DP is defined as follows.

Definition 5.1 (Dirichlet Process). *Let* $(\Omega, \mathcal{F})$ *be a measurable space, and* μ *be a finite measure over* Ω. *A* Dirichlet process *with base measure* μ, *denoted by* $\mathrm{DP}(\mu)$, *is defined to be a distribution of random probability measure* D *over* Ω, *such that for*

any finite measurable partition $(A_1, \ldots, A_n)$*, the random vector* $(D(A_1), \ldots, D(A_n))$ *has a Dirichlet distribution:*

$$(D(A_1), \ldots, D(A_n)) \sim \mathrm{Dir}(\mu(A_1), \ldots, \mu(A_n)). \tag{5.12}$$

Note that in other literatures, μ is often factorized into a *base distribution* $p_\mu \triangleq \mu/\mu(\Omega)$, and a *concentration parameter* $\alpha_\mu \triangleq \mu(\Omega)$. For the convenience of later discussion, we use the base measure μ as a whole to characterize a DP in this thesis.

According to *Kolmogorov's extension theorem* [55], there exists a unique stochastic process that satisfies the property above, meaning that a base measure μ corresponds uniquely to a Dirichlet Process. The definition above is a descriptive definition. Actually, we can also explicitly construct a DP through an interesting process called *stick breaking*, which will be presented later.

From a DP, each sample path D itself is a probability measure over Ω, and therefore $\mathrm{DP}(\mu)$ is essentially a *distribution over distributions*. Such a stochastic process – one of which every sample is a probability measure – is called a *random probability measure*. There is a very important property that distinguishes DP from other random probability measures – *neutrality*, which is defined by the proposition below.

Proposition 5.1. *Let* μ *be a measure over* Ω*,* $D \sim \mathrm{DP}(\mu)$ *and* A, B, C *be disjoint measurable subsets of* Ω*. Then,* $(D(A), D(B), D(C))$ *is a neutral vector, meaning that the ratio* $D(A)/D(B)$ *is independent of the value of* $D(C)$*.*

There are other important properties about a DP. We will show later that D is almost surely a discrete distribution over a countably infinite subset of Ω, which can be established via the stick breaking construction. Previous work mostly uses two methods to construct DPs: the *Pólya urn scheme* and the *stick-breaking construction*, which we will describe in what follows.

5.2.2 Pólya Urn and Chinese Restaurant Process

Consider a generative model: $D \sim \mathrm{DP}(\mu)$ and $\theta_1, \ldots, \theta_n | D \sim D$. The Pólya urn scheme is a sequential procedure to directly draw $\theta_1, \ldots, \theta_n$ from this generative

model, without making D explicit. Given $\theta_1, \ldots, \theta_{i-1}$, by integrating out D, we have

$$\theta_i|\theta_1, \ldots, \theta_{i-1} \sim \sum_{j=1}^{i} \frac{1}{\alpha_\mu + (i-1)}\delta_{\theta_i} + \frac{\alpha_\mu}{\alpha_\mu + (i-1)}p_\mu. \tag{5.13}$$

We can see that θ_i has a positive probability repeating a value that was seen earlier. The more often that an atom value has been sampled, the more likely that it is to be repeated in successive draws. As the number of samples n increases, the number of distinct values also increases, but at a much slower rate (about $O(\log(n))$). This is known as the *clustering property*. To make this clear we introduce the terminology *atoms* to refer to such distinct values, denoted by $\phi_1, \phi_2, \ldots$, and thus rewrite Eq.(5.13) into

$$\theta_i|\theta_1, \ldots, \theta_{i-1} \sim \sum_{k=1}^{K_{i-1}} \frac{m_{i-1}^{(k)}}{\alpha_\mu + (i-1)}\delta_{\phi_k} + \frac{\alpha_\mu}{\alpha_\mu + (i-1)}p_\mu. \tag{5.14}$$

Here, K_{i-1} is the number of distinct atoms in the first $i-1$ samples, and $m_{i-1}^{(k)}$ is the number of appearances of the atom ϕ_k. Moreover, we don't actually have to explicitly maintain the samples $\theta_1, \ldots, \theta_n$; instead, we can use a label z_i to indicate which atom the i-th sample takes value from, *i.e.* $\theta_i = \phi_{z_i}$. Using *atoms* and *labels*, each with its own distinctive meaning, we can sample θ_i as follows

1. Draw $z_i \in \{1, \ldots, K_{i-1} + 1\}$ conditioned on $z_1, \ldots, z_{i-1}$ with

$$\Pr(z_i = k|z_1, \ldots, z_{i-1}) = \begin{cases} m_{i-1}^{(k)}/(\alpha_\mu + (i-1)) & (k \in \{1, \ldots, K_{i-1}\}), \\ \alpha_\mu/(\alpha_\mu + (i-1)) & k = K_{i-1} + 1. \end{cases} \tag{5.15}$$

2. If $z_i \leq K_{i-1}$, we are done, and it simply means that $\theta_i = \phi_{z_i}$; otherwise, we draw a new atom $\phi_{K_{i-1}+1}$ from the base distribution p_μ, and set $z_i = K_{i-1} + 1$.

Chinese Restaurant Process

This procedure is closely related to the *Chinese restaurant process*, as described below. Consider a Chinese restaurant with an infinite number of circular tables, each of infinite capacity. When a new customer enters, he chooses a table to sit at. The

probability that a particular table is chosen is proportional to the number of people already seated there. There is also a positive probability proportional to α_μ that a new table is selected. Using z_i to indicate the index of the table that the i-th customer chooses, we will get a sequence $z_1, \ldots, z_n$, when n customers have been seated. Comparison of this process with the Pólya urn scheme suggests that the generation model of such sequence can be exactly characterized by Eq.(5.15). Consequently, the sampling procedure can be reformulated equivalently as

$$\begin{aligned} (z_1, \ldots, z_n) &\sim \mathrm{CRP}(\alpha_\mu); \\ \phi_k &\sim p_\mu; \qquad \text{for } k = 1, \ldots, K_n. \end{aligned} \tag{5.16}$$

Here, we first draw the labels from the Chinese restaurant process as described above, which we denote by $\mathrm{CRP}(\alpha_\mu)$. This labeling process results in clusters of data, each corresponding to a label. Then, an atom ϕ_k, the parameter of the component to explain the samples classified to the k-th table, can be independently drawn from p_μ for each cluster.

Exchangeability

The sequential dependency as in Eq.(5.13) might seem to indicate that the joint probability of $\theta_1, \ldots, \theta_n$ depends on the their order. However, this is not really the case.

Consider the Chinese Restaurant Process, it is not difficult to derive the joint probability of the labels as

$$p_{\mathrm{CRP}}(z_1, \ldots, z_n | \alpha_\mu) = \prod_{i=2}^{n} p(z_i | z_1, \ldots, z_{i-1}) = \frac{\prod_{k=1}^{K_n} \alpha (m_n^{(k)} - 1)!}{\prod_{i=0}^{n-1} (\alpha + i)}. \tag{5.17}$$

The right hand side of this equation can be obtained by rearranging the factors.

Moreover, with the atom parameters taken into account, we further get

$$p(\theta_1, \ldots, \theta_n|\mu) = p(z_1, \ldots, z_n; \phi_1, \ldots, \phi_{K_n}|\mu)$$
$$= p_{\text{CRP}}(z_1, \ldots, z_n|\alpha_\mu) \prod_{k=1}^{K_n} p_\mu(\theta_k). \tag{5.18}$$

It is clear from Eq.(5.17) that $p_{\text{CRP}}(z_1, \ldots, z_n)$ only depends on the number of times each label occurs, but not on their order. As a result, the joint probability of $\theta_1, \ldots, \theta_n$ does not depend on the order either, implying that $\theta_1, \ldots, \theta_n$ is an *exchangeable sequence.*

De Finetti's theorem [55] states that exchangeable samples are conditionally independent given some latent variable. Here, the latent variable is D, a DP sample path that serves as the prior of atoms.

5.2.3 Stick-breaking Construction

The stick breaking construction proposed by Sethuraman [90] provides a direct construction of DP. Let $\{\beta_k\}_{k=1}^{\infty}$ be a sequence of independent random variables from a beta distribution, as

$$\beta_k \sim \text{Beta}(1, \alpha_\mu), \qquad \text{for } k = 1, 2, \ldots, \tag{5.19}$$

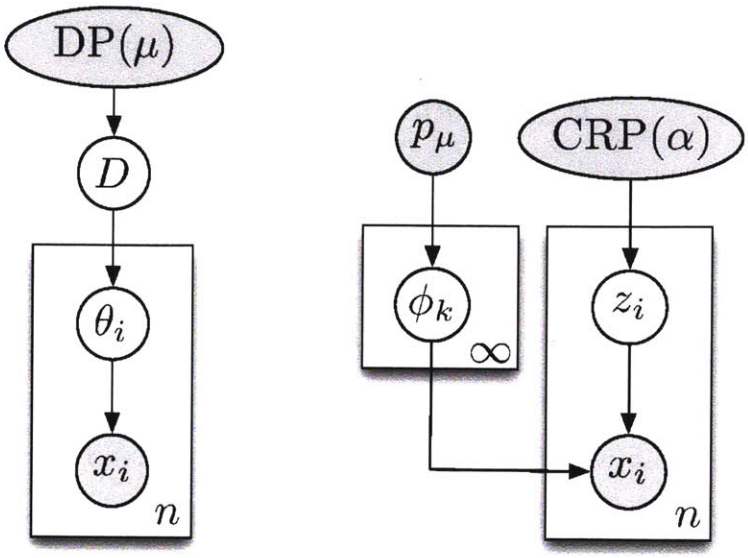
and $\{\phi_k\}_{k=1}^{\infty}$ be a sequence of independent variables from the base distribution p_μ, as

$$\phi_k \sim p_\mu, \qquad \text{for } k = 1, 2, \ldots. \tag{5.20}$$

Then, we can construct a random series D as

$$\pi_k = \beta_k \prod_{l=1}^{k-1}(1 - \beta_l), \qquad \text{and} \qquad D = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k}. \tag{5.21}$$

Intuitively, this sampling procedure is analogous to a stick breaking process. Starting from a stick of unit length, we first draw a random fraction $\beta_1 \sim \text{Beta}(1, \alpha_0)$, and let



(a) Basic formulation (b) The formulation using CRP

Figure 5-5: This figure shows the graphical representations of the DP mixture model. **(a)** Basic formulation: each sample is associated with a parameter, which is generated from an underlying DP sample D. **(b)** An equivalent formulation derived based on the Chinese restaurant process. Here, an infinitely pool of atoms is independently generated, and each sample is attached a label drawn from a CRP. This labels associates the sample with an atom chosen from the pool.

$\pi_1 = \beta_1$. For the remaining part, whose length is $1 - \pi_1$, we draw β_2 to further break it into two parts, and let $\pi_2 = \beta_2(1 - \pi_1)$. By recursively applying these steps, we end up with an infinite sequence $(\pi_k)_{k=1}^{\infty}$, which we call the *stick breaking coefficients.* In addition, the expression given in Eq.(5.21) is called the *stick breaking representation.*

Sethuraman showed that D obtained in this way is a random probability measure that has $D \sim \mathrm{DP}(\mu)$. The stick breaking construction not only offers a useful representation of DP samples, but also establishes an important fact that a Dirichlet process is almost surely a discrete distribution over a countably infinite set.

5.2.4 DP Mixture Models

The discrete nature of a Dirichlet process together with the clustering property of its samples makes it a useful prior for the components in a mixture model. Generally, a

DP mixture model is formulated as

$$\begin{aligned} D &\sim \mathrm{DP}(\mu); \\ \theta_i|D &\sim D, \qquad \text{for } i = 1, \ldots, n; \\ x_i|\theta_i &\sim \mathcal{G}(\theta_i), \qquad \text{for } i = 1, \ldots, n. \end{aligned} \tag{5.22}$$

Here, $\mathrm{DP}(\mu)$ is a Dirichlet process over the parameter space Ω, with base measure μ, and $\mathcal{G}(\theta_i)$ is a generative model with parameter θ_i. The graphical representation is shown in Figure 5-5(a). In this model, we assume that the parameters of all component models come from D, which itself is a distribution over Ω. Then, to generate x_i, we first draw the corresponding component parameter θ_i from D, and then draw $x_i \sim L(\theta_i)$. Using the Chinese restaurant process, we can rewrite the generation process as

$$\begin{aligned} \phi_k &\sim p_\mu; \qquad \text{for } k = 1, 2, \ldots; \\ (z_1, \ldots, z_n) &\sim \mathrm{CRP}(\alpha_\mu); \\ x_i &\sim \mathcal{G}(\phi_{z_i}), \qquad \text{for } i = 1, \ldots, n. \end{aligned} \tag{5.23}$$

The graphical model based on this formulation is shown in Figure 5-5(b). Unlike a finite mixture model, where labels are chosen from a finite set, a DP mixture model provides an infinite collection of components – though not all of them are made explicit. Moreover, labels are generated from a Chinese restaurant process, which ensures positive chances to create new labels. Therefore, the number of components is no longer needed to be fixed prior to estimation, as the inference algorithm will determine this, introducing new components when they are needed.

5.3 Dependent Dirichlet Processes

The formulation above implicitly assumes that all observations are exchangeable, which, however, is not always true in practice, especially in a dynamic context. Con-

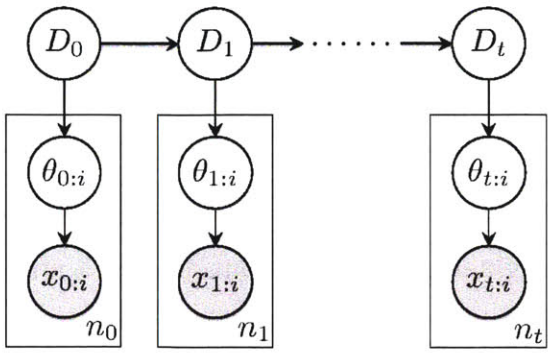


Figure 5-6: This figure shows an extended DP mixture model, which incorporates temporal dependency between DPs at consecutive time. In this model, there is a DP mixture model at each time step. Based on the temporal dependency between them, the DPs together form a Markov chain. Conditioned on the DP prior at time t, the model parameters $\theta_{t:1}, \ldots \theta_{t:n_t}$ and thus the observations $x_{t:1}, \ldots, x_{t:n_t}$ are independently generated.

sider a scenario where we have observations at different time steps. Whereas it is reasonable to assume exchangeability within each time step, extending such assumption across different time steps is often inappropriate, as the underlying generative models can evolve over time.

Under a dynamic setting, it is a natural idea to introduce different DP priors for different time steps. In particular, suppose there are $(T+1)$ time steps: $0, \ldots, T$. We denote the observations at time t by $X_t = \{x_{t:1}, \ldots, x_{t:n_t}\}$, and the corresponding component parameters by $\theta_{t:1}, \ldots, \theta_{t:n_t}$. Then, the generative model can be written as

$$\begin{aligned} D_t &\sim \mathrm{DP}(\mu); \\ \theta_{t:i}|D_t &\sim D_t \qquad \text{for } i = 1, \ldots, n_t, \quad t = 0, \ldots, T; \\ x_{t:i}|\theta_{t:i} &\sim \mathcal{G}(\theta_{t:i}), \qquad \text{for } i = 1, \ldots, n_t, \quad t = 0, \ldots, T. \end{aligned} \tag{5.24}$$

In practical problems, there often exist strong dependencies between the generative models at different time steps. The primary goal of this chapter is to develop a stochastic process of DPs, as shown in Figure 5-6, and thus provide a generic methodology to incorporate such temporal dependencies. Particularly, we aim to develop a

process that supports dynamic creation and removal of components, as well as variation of component parameters, while maintaining the desirable property that the marginal distribution of D_t remains a DP.

In the parametric world, one may often easily find a way to transform a parameter that preserves its original form of distribution, *e.g.* a noisy linear transform of a Gaussian distributed variable remains Gaussian. However, this becomes a challenging issue when we are working with Dirichlet processes. As we shall see, whereas there have been a lot of efforts devoted to the incorporation of statistical dependency between DPs, previous methods are limited both theoretically and practically.

Later in this chapter, we will introduce a new perspective of Dirichlet processes by examining their connections with Poisson processes, and thereon derive a new construction of dependent DPs that allows a variety of transformations on DPs while retaining the marginal distributions as DPs.

5.3.1 A Brief Review

We first briefly review previous work on dependent Dirichlet processes. The notion of *dependent Dirichlet process (DDP)* was coined by MacEachern in his pioneering work [66]. In particular, he discussed a special form of DDP, called *single-p DDP*, where each component distribution is extended to a stochastic process that describes how the component parameter evolves with the covariate. Whereas several extensions were also discussed, such as the use of non-stationary stochastic processes for describing atom evolution, this work did not other forms of variations, *e.g.* dynamic incorporation of new atoms.

Following this work, a variety of DDP constructions have been developed, which mainly fall into three categories.

Convex Combinations

A straightforward way to extend the Dirichlet processs is to generate D_t, the nonparametric mixture with covariate t, as a convex combination of an existing DP and

a new one. Müller *et al.* [72] proposed a method that combines the inference across related nonparametric mixture models through a convex combination as below

$$D_t = \epsilon F_0 + (1 - \epsilon) F_t. \tag{5.25}$$

Here, F_0 and F_t are independent sample paths from a DP, which respectively capture the common atoms across all groups and the atoms only used by the t-th group.

Zhu *et al.* [120] proposed a time sensitive DP mixture model, which assigns each atom a time-dependent weight, defined as follows

$$w(t, j) = \sum_{i|t_i<t, s_j=j} k(t - t_i). \tag{5.26}$$

This is a variant of the standard Chinese restaurant process, where the contribution of each occurrence of an atom attenuates over time. Such a weighting strategy is actually equivalent to introducing a sequence of latent DPs, each for a time step, and considering the mixture at each time step as their moving average.

Caron *et al.* [17] directly generalized the Pólya urn scheme, where a subset of balls from the urn is randomly chosen and deleted. With an appropriate choice of deletion probability, this would result in a stationary process, where existing atoms have geometrically distributed life spans, and new atoms constantly come up. The generalized Pólya urn model was later applied for spike sorting [35].

Ahmed and Xing [1] developed a temporal DP mixture model called Recurrent Chinese restaurant process. Here, each atom occurring at previous time step can be chosen with the following probability

$$\frac{n_{k,t-1} + n^i_{k,t}}{N_{t-1} + (i - 1) + \alpha}. \tag{5.27}$$

Here, $n_{k,t-1}$ is the number of occurrences of atom k at $t - 1$, $n^{(i)}_{k,t}$ is the number of occurrences at t up to the $(i - 1)$-th sample, N_{t-1} is the number of data samples at $t - 1$. Whereas this generalization was motivated from an algorithmic standpoint, it can be considered to be resulted from a convex combination of the previous DP

sample and a new one.

Despite the technical differences, the methods in this category have an important aspect in common: they consider a dependent DP as a combination of two or more source DPs. Such a combination is either formulated directly or through an algorithmic design that maintains a positive chance to draw atoms occurring at previous time steps. Such constructions can generally be expressed as follows

$$D_t = \sum_{s \in S_t} \beta_{s,t} H_s. \tag{5.28}$$

Here, S_t denotes the set of DPs that contributes to D_t, and H_s can be a DP sample at previous time step or a latent one. This type of formulation has an important problem: the marginal distribution of D_t is no longer a DP in general. Moreover, other issues, like how to achieve a balance between inheritance (*i.e.* using old atoms) and innovation (*i.e.* introducing new atoms), have not been completely addressed.

Permutation of Stick Breaking Terms

Another approach to constructing dependent DPs is based on the stick breaking representation introduced by Sethuraman [90]. Griffin and Steel [40] proposed an interesting construction called πDDP, where each dependent DP is derived through a permutation of stick breaking terms. Recall that the stick breaking representation of a DP is given by

$$D = \sum_{i=1}^{n} c_i \delta_{\theta_i}, \quad \text{with } c_i = v_i \prod_{j=1}^{i-1}(1 - v_j), \quad v_i \sim \text{Beta}(1, \alpha). \tag{5.29}$$

Note that here any permutation of the stick breaking coefficients $(v_i)_{i=1}^{\infty}$ still results in a DP, as

$$D_t = \sum_{i=1}^{n} c_i^{(t)} \delta_{\theta_i}, \quad \text{with } c_i^{(t)} = v_{\pi_t(i)} \prod_{j=1}^{i-1}(1 - v_{\pi_t(j)}). \tag{5.30}$$

Here, π_t is a permutation. Clearly, all these permutated versions share the same collection of atoms, but with different weights assigned to them. Thus, they are

related to each other. This paper also proposed a useful way to generate the ordering. The basic idea is to sample the arrival order from a point process.

Building upon this idea, Chung and Dunson [19] proposed the local DP model (lDP). This work aims to formulate a joint distribution over a collection of DPs, and, for this purpose, introduces a universal pool of atoms. Each DP under this joint formulation contains a subset of atoms in the pool, and is associated with a covariate-dependent permutation.

These methods guarantees that the marginal distribution of each resultant process remains a DP. However, they bring about another nontrivial question: how to devise the process that generate the permutation such that it can express the desired dependency structure. Also, as such formulations involve random permutations, the sampling schemes tend to be very complicated, often leading to considerably increased demand of computational costs.

Hierarchical Dirichlet Process

Hierarchical Dirichlet Process (HDP) was proposed by Teh *et al.* [100], which has become one of the most widely used approaches to constructing dependent nonparametric mixture models. Taking advantage of the fact that each sample of a DP is a distribution over the underlying space, this model organizes DP samples into a tree, where parents serve as the base probability measure of their children. The generative formulation is given below

$$D_0 \sim \mathrm{DP}(\alpha, B), \quad D_t \sim \mathrm{DP}(\gamma, D_0), \quad \text{for } t = 1, 2, \ldots. \tag{5.31}$$

This way ensures that the atoms in a child DP are from its parent, thus offering a way for atoms to be shared across different children.

Ren *et al.* [82] applied this idea to a dynamic context, and developed the dynamic HDP (dHDP) to model sequential data. The generative formulation of dHDP is

$$D_t = (1 - w)D_{t-1} + wH_t, \quad H_t = \mathrm{DP}(\alpha, D_0), \quad D_0 \sim \mathrm{DP}(\gamma, B). \tag{5.32}$$

This model adopts the convex combination formula to evolve a DP over time, where the innovative process is generated from an HDP.

In [100], Teh *et al.* discussed an extension called HDP-HMM, which is based on a Markov chain with infinitely many states. In this model, the distribution of the next state conditioned on any current state is a child DP. All such DPs share the same parent. Hence, the transition is between the atoms from a common parent DP. Fox *et al.* [31] further extends this formulation by introducing an additional sticky variable to control the self-transition bias.

Kim and Smyth [53] extended HDP along a different direction. They introduced a random perturbation for each group, called *random effects*, which allows the parameters to each atom to vary when inherited by child DPs.

HDP is very useful in many applications that involve groups of data. However, HDP and its variants are subject to a fundamental limitation, that is, the DPs must be organized into a tree-structure.

Spatially Normalized Gamma Process

Recently, Rao and Teh [78] proposed a new way for dependent DP construction, called *spatially normalized gamma process*, which allows more flexible configuration of the dependency structure. This formulation leverages an important relation between Gamma and Dirichlet process, that is, normalizing a Gamma process results in a DP.

In particular, it defines a gamma process G over an extended space. For each group t, a DP D_t is derived through normalized restriction of G to a measurable subset. The DPs derived on overlapping subsets are thus dependent. This construction can be reduced to the following form:

$$D_t = \sum_{j \in R_t} c_{tj} H_j, \quad \text{with} \quad (c_{tj})_{j \in R_t} \sim \mathrm{Dir}((\alpha_j)_{j \in R_t}). \tag{5.33}$$

Here, R_t is the subset of latent DPs used for D_t, and the coefficients c_{tj} are random variables from a Dirichlet distribution.

It is worth noting that under this formulation, the relative weights of two latent

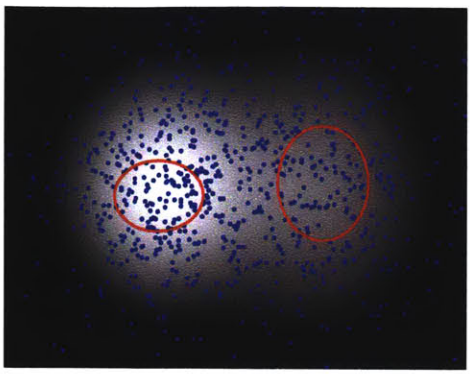


Figure 5-7: This figure shows a realization of a Poisson process whose base measure μ is inhomogeneous over the underlying space, which is a collection of points. Let A and B denote the two regions marked by red ellipses. Then $N_\Pi(A)$ and $N_\Pi(B)$ are respectively the numbers of points therein, which are are independent variables.

sources are restricted to be the same in all groups that inherit from both. Also, it does not provide a way for atom parameters to vary across groups.

5.3.2 Poisson, Gamma, and Dirichlet Processes

Our construction of *dependent* Dirichlet processes relies on the theoretical connections between Poisson, Gamma, and Dirichlet processes, as well as an important probabilistic property called *complete randomness*. In this section, we provide a brief review of these concepts, and discuss the relations between them.

In general, a Poisson process Π is a random point process[1], *i.e.* a stochastic process of which each realization is a set of points. As illustrated in Figure 5-7, each realization of Π uniquely induces a counting measure N_Π over Ω, given by $N_\Pi(A) \triangleq \#(\Pi \cap A)$ for each $A \in \mathcal{F}$. Formally, a Poisson process is defined as below.

Definition 5.2 (Poisson process). *Let $(\Omega, \mathcal{F})$ be a measurable space, and μ be a σ-finite measure over Ω. Let Π be a random point process on Ω, then it is called a* Poisson process *with mean measure μ, denoted by $\Pi \sim \mathrm{PP}(\mu)$, if its induced counting measure N_Π satisfies the following two properties:*

[1]In this work, we are considering generic Poisson processes (also known as spatial Poisson process) that can be defined over any measurable space, instead of the temporal Poisson processes that are often discussed in introductory textbooks.

1. *for any measurable subset $A \in \mathcal{F}$, the random variable $N_\Pi(A)$ is Poisson distributed, as $N_\Pi(A) \sim \text{Poisson}(\mu(A))$;*

2. *given a collection of disjoint subsets $A_1, \ldots, A_n \in \mathcal{F}$, the variables $N_\Pi(A_1), \ldots, N_\Pi(A_n)$ are independent.*

The second property here is referred to as *complete randomness*, which is the key aspect that distinguishes Poisson processes from other point processes. Generally, complete randomness is defined to be a property possessed by a special family of stochastic processes, called *complete random measures*:

Definition 5.3 (Completely random measure). *A random measure M over measurable space $(\Omega, \mathcal{F})$, i.e. a stochastic process whose realizations are measures, is called* completely random *if given any collection of disjoint measurable sets $A_1, \ldots, A_n \in \mathcal{F}$, the random variables $M(A_1), \ldots, M(A_n)$ are independent.*

In particular, a random point process Π is called a a completely random point process *if its induced counting measure N_Π is a completely random measure.*

It is clear that each Poisson process is a completely random point process. More importantly, under mild technical conditions, the converse also holds. To be more specific, *Poisson processes are the only point processes that are completely random*, as stated by the following theorem.

Theorem 5.1. *A random point process Π on a regular measurable space is a Poisson process if and only if it is completely random.*

This theorem suggests that if we can construct a point process on a regular space such that it is completely random, then it is guaranteed that the resultant process is a Poisson process. This is one of the key ideas behind our construction, as we shall see in next section.

Next, we describe how a Dirichlet process can be derived from a Poisson process, via another important stochastic process, called *Gamma process*. Consider a Poisson process Π^* over a product space $\Omega \times \mathbb{R}^+$, where each point can be expressed as a pair (θ, w_θ) with $\theta \in \Omega$ and $w \in \mathbb{R}^+$. Intuitively, we can consider θ as a parameter, and w_θ

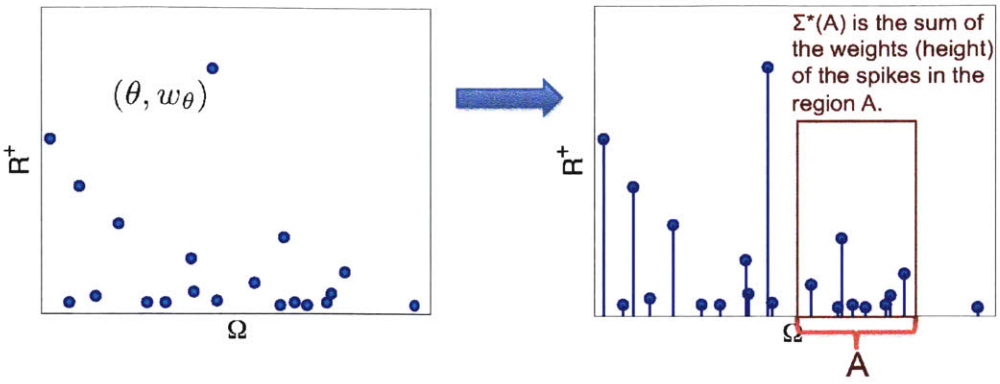


Figure 5-8: This figure illustrates how a Gamma process can be constructed from a Poisson process over a product space. On the left shows a realization of a Point process Π^* over the product space $\Omega \times \mathbb{R}^+$, where each point is a pair (θ, w_θ). Converting each such point to a term $w_\theta \delta_\theta$ and combining them to form a series, we obtain a random measure as in Eq.(5.34). In particular, when $\Pi^* \sim \mathrm{PP}(\mu \times \gamma)$, Σ^* is a Gamma process that has $\Sigma^* \sim \Gamma\mathrm{P}(\mu)$.

as the weight associated with it. Given a realization of Π^*, we can define a measure G over Ω as

$$G(A) = \sum_{\theta \in \Pi^* \cap A} w_\theta, \qquad A \in \mathcal{F} \text{ is a measurable subset of } \Omega. \tag{5.34}$$

Intuitively, $G(A)$ sums up the weights w_θ associated with the parameters θ that fall in A, as illustrated in Figure 5-8. Since Π^* is random, G is actually a random measure.

The following lemma further shows that G is completely random, and under a special choice of the mean measure of Π^*, it is exactly a Gamma process.

Lemma 5.1. *Suppose $\Pi^* \sim \mathrm{PP}(\mu^*)$ be a Poisson process over the product space $\Omega \times \mathbb{R}^+$, then G as defined in Eq.(5.34) is a completely random measure. In particular, when $\mu^* = \mu \times \gamma$ with $\gamma(dw) = w^{-1}e^{-w}dw$, we have for each measurable subset A,*

$$G(A) \sim \mathrm{Gamma}(A), \tag{5.35}$$

which implies that G is a Gamma process with base measure μ, denoted by $G \sim \Gamma\mathrm{P}(\mu)$.

Furthermore, as stated by the lemma below, normalizing each sample path of a

Gamma process gives rise to a Dirichlet process.

Lemma 5.2. *Let $G \sim \Gamma\mathrm{P}(\mu, \lambda)$ be a Gamma process over a measurable space $(\Omega, \mathcal{F})$, and μ is a finite measure. Let $D = G/G(\Omega)$, then*

$$D \sim \mathrm{DP}(\mu), \qquad \textit{and} \qquad G(\Omega) \sim \mathrm{Gamma}(\mu(\Omega)). \tag{5.36}$$

In addition, D is independent from $G(\Omega)$.

Specifically, as in Eq.(5.34), each sample path G of a Gamma process can be written in form of a series as

$$G = \sum_{\theta \in \mathcal{D}(\Pi^*)} w_\theta \delta_\theta, \qquad \mathcal{D}(\Pi^*) \triangleq \{\theta : (\theta, w_\theta) \in \Pi^*\}. \tag{5.37}$$

When μ is finite, it is absolutely convergent, and equals $G(\Omega)$. Normalizing G, we get

$$D = G/G(\Omega) = \sum_{\theta \in \mathcal{D}(G)} \tilde{w}_\theta \delta_\theta, \quad \text{with } \tilde{w}_\theta = w_\theta / G(\Omega). \tag{5.38}$$

This is the stick breaking representation of D, and the normalized weights $\tilde{w}_\theta$ are the stick breaking coefficients. Lemma 5.1 and Lemma 5.2 together reveal the inherent connections between Poisson, Gamma, and Dirichlet processes, and suggest that *we can construct a Dirichlet process over Ω based on a Poisson process over the product space $\Omega \times \mathbb{R}^+$.*

Next, we consider a procedure in the opposite direction, namely obtaining the underlying Poisson process of a given Dirichlet process. Suppose we have a Dirichlet process D over a measurable space Ω, and its stick breaking representation is

$$D = \sum_{i=1}^{\infty} \pi_i \delta_{\theta_i}. \tag{5.39}$$

As an intermediate step, we first construct a Gamma process. Specifically, we inde-

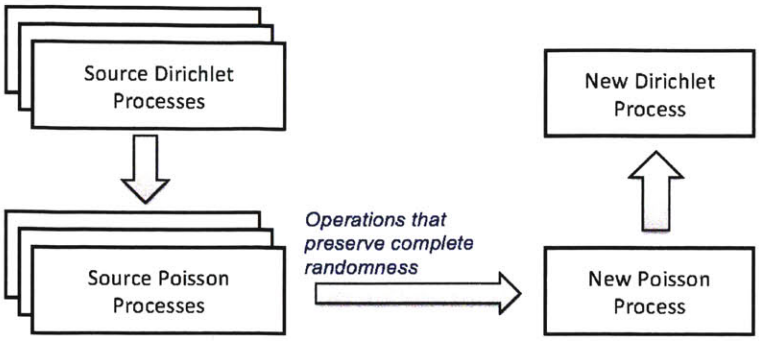


Figure 5-9: This diagram shows the high-level idea behind our approach. Rather than directly working with the DPs directly, we do the construction in the Poisson domain, obtaining a new Poisson process via the operations that preserve complete randomness. Then, we can derive a DP from the resultant Poisson process, based on their intrinsic connections.

pendently draw a Gamma distributed variable $g \sim \text{Gamma}(\mu(\Omega))$, and let

$$G = gD = \sum_{i=1}^{\infty} g\pi_i \delta_{\theta_i}. \tag{5.40}$$

Then G is a Gamma process over Ω, as $G \sim \Gamma\text{P}(\mu)$. Based on the relations between Gamma and Poisson processes, we can then obtain the underlying Poisson process over $\Omega \times \mathbb{R}^+$:

$$\Pi = \{(\theta_i, g\pi_i) : i = 1, 2, \ldots\} \sim \text{PP}(\mu \times \gamma). \tag{5.41}$$

It is easy to verify that the Dirichlet process derived from Π is exactly D. For the convenience of later discussion, we denote the Poisson process Π as obtained above by $\mathcal{P}^*(D; g)$.

5.3.3 Poisson-based Construction of DDP

This section presents our approach to constructing *dependent Dirichlet process* – a DP with probabilistic dependency on others. As depicted in Figure 5-9, the basic idea of this approach consists of two related aspects:

1. Exploiting the inherent connection between Poisson and Dirichlet processes as introduced in previous section, we can construct dependent DPs via the construction of *dependent Poisson processes*. Specifically, given a set of source DPs

over a space Ω, we first obtain their underlying Poisson processes over $\Omega \times \mathbb{R}^+$, and thereon construct a dependent Poisson process, from which a new DP can be derived based on the Poisson-Dirichlet relations.

2. This strategy reduces the problem to the one of constructing dependent Poisson processes. To solve this problem, we develop a generic methodology that rests upon the notion of *complete randomness*. The idea is that by applying operations that can preserve complete randomness to a set of Poisson processes, we get a new point process that is completely random, which, by Theorem 5.1, remains a Poisson process.

In what follows, we consider three such operations: *superposition*, *subsampling*, and *point transition*. When applied to the underlying Poisson processes, they produce new Poisson processes and thus new Dirichlet processes. These operations together offer great flexibility for evolving a DP dynamically.

Superposition

An important way to create dependent Dirichlet process is to combine existing ones. Whereas it is a natural idea to simply combine them through convex combination, the resultant process is, however, no longer a DP in general. In this work, we take a new approach – combine DPs through the union of the underlying Poisson processes.

Given a collection of independent DPs $D_1, \ldots, D_m$. Following the procedure described in previous section, we independently draw m Gamma distributed coefficients: $g_1, \ldots, g_m$, with $g_k \sim \text{Gamma}(\mu_k(\Omega))$, and obtain their underlying Poisson processes, as

$$\Pi_k^* = \mathcal{P}^*(D_k; g_k) \;\sim\; \text{PP}(\mu_k \times \gamma). \tag{5.42}$$

The union of these Poisson processes is $\Pi_{sup}^* = \Pi_1^* \cup \cdots \cup \Pi_m^*$. The following theorem establishes the fact that Π_{sup}^* remains a Poisson process.

Theorem 5.2 (Superposition Theorem of Poisson Processes). *Let* $\Pi_1, \ldots, \Pi_m$ *be in-*

dependent Poisson processes on Ω *with* $\Pi_k \sim \mathrm{PP}(\mu_k)$*, then their union has*

$$\Pi_1 \cup \cdots \cup \Pi_m \ \sim \ \mathrm{PP}(\mu_1 + \cdots + \mu_m). \tag{5.43}$$

This immediately follows that

$$\pi^*_{sup} \sim \mathrm{PP}((\mu_1 + \cdots + \mu_m) \times \gamma). \tag{5.44}$$

According to the relations between Poisson and Gamma processes, Π^*_{sup} corresponds to a Gamma process, denoted by G_{sup}, which equals the sum[2] of the ones associated with $\Pi^*_1, \ldots, \Pi^*_m$:

$$G_{sup} = \sum_{(\theta, w_\theta) \in \Pi^*_1 \cup \cdots \cup \Pi^*_m} w_\theta \delta_\theta = \sum_{k=1}^{m} \sum_{(\theta, w_\theta) \in \Pi_k} w_\theta \delta_\theta = \sum_{k=1}^{m} G_k = \sum_{k=1}^{m} g_k D_k. \tag{5.45}$$

Normalizing G_{sup}, we finally get a Dirichlet process D_{sup} as follows.

$$D_{sup} = G_{sup}/G_{sup}(\Omega) = \sum_{k=1}^{m} \frac{g_k}{g_{sum}} D_k = \sum_{k=1}^{m} \alpha_k D_k \ \sim \ \mathrm{DP}(\mu_1 + \cdots + \mu_m). \tag{5.46}$$

Here, $g_{sum} = g_1 + \cdots + g_m$, and $\alpha_k = g_k / g_{sum}$. We note that the coefficients here are Dirichlet distributed random variables, as $(\alpha_1, \ldots, \alpha_m) \sim \mathrm{Dir}(\mu_1(\Omega), \ldots, \mu_m(\Omega))$. Consequently, *one can construct a Dirichlet process through a random convex combination of independent Dirichlet processes*, as

$$D_{sup} = D_1 \oplus \cdots \oplus D_m \triangleq c_1 D_1 + \cdots + c_m D_m \ \sim \ \mathrm{DP}(\mu_1 + \cdots + \mu_m). \tag{5.47}$$

For convenience, we use $D_1 \oplus \cdots \oplus D_m$ to denote the DP resulted from the superposition of $D_1, \ldots, D_m$. The procedure described above illustrates in detail how one can construct a dependent Dirichlet process via the operations on Poisson processes that preserve complete randomness. In regard to this procedure, we have the following

[2]Here, we utilize a fact that the realizations of independent Poisson processes are almost surely disjoint, and consequently the sum of the terms in the union is almost surely equal to the sum of individual series.

discussions.

1. Although playing a crucial role in this theoretical derivation, the underlying Poisson processes, however, do not need to be explicitly instantiated in the actual construction as given by Eq.(5.47).

2. That the coefficients $c_1, \ldots, c_m$ are drawn from a Dirichlet distribution distinguishes our construction from the simple convex combinations that use deterministic coefficients. This is the key to make D_{sup} a Dirichlet process.

Subsampling

We can also construct a dependent DP by extracting a portion of a given DP. This is done by a complete randomness preserving operation called *random subsampling* – a special case of *random coloring*, which is described below.

Theorem 5.3 (Coloring Theorem of Poisson Processes). *Let $\Pi \sim \mathrm{PP}(\mu)$ be a Poisson process over a measurable space Ω, $\mathcal{C}$ be a finite subset. Each $c \in \mathcal{C}$ is associated with a measurable function $q_c : \Omega \to [0,1]$, such that for each $\theta \in \Omega$, $\sum_{c\in\mathcal{C}} q_c(\theta) = 1$. For each $\theta \in \Pi$, we draw a discrete variable $z_\theta \in \mathcal{C}$, with $\Pr(z_\theta = c) = q_c(\theta)$. Let $\Pi_c = \{\theta \in \Pi : z_\theta = c\}$ for each $c \in \mathcal{C}$, then Π_c is a Poisson process as $\Pi_c \sim \mathrm{PP}(\mu_c)$ with $\mu_c = q_c\mu$, i.e. $\mu_c(d\theta) = q_c(\theta)\mu(d\theta)$, and Π_c for different values of c are independent.*

The procedure in this theorem can be intuitively interpreted as follows. Given a realization of a Poisson process Π, we randomly assign a color $c \in \mathcal{C}$ to each point, according to the distribution over $\mathcal{C}$ given by $(q_c(\theta))_{c\in\mathcal{C}}$. Then we divide the entire set of points into $|\mathcal{C}|$ subsets depending on the color associated with each point. In this way, we get $|\mathcal{C}|$ point processes. Theorem 5.3 suggests that they are independent Poisson processes, and particularly, $\Pi_c \sim \mathrm{PP}(q_c\mu)$ for each $c \in \mathcal{C}$. Restricting $\mathcal{C}$ to be $\{0,1\}$, we have:

Corollary 5.1. *Let $\Pi \sim \mathrm{PP}(\mu)$ be a Poisson process Ω, and $q : \Omega \to [0,1]$ be a measurable function. If we independently draw $b_\theta \in \{0,1\}$ for each $\theta \in \Pi$ with $\Pr(b_\theta = 1) = q(\theta)$, and let $\Pi_{sub} = \{\theta \in \Pi : b_\theta = 1\}$, Then Π_{sub} and Π/Π_{sub} are independent Poisson processes over Ω, with $\Pi_{sub} \sim \mathrm{PP}(q\mu)$, and $\Pi/\Pi_{sub} \sim \mathrm{PP}((1-q)\mu)$.*

The procedure described above is essentially a *random subsampling* process, where we determines whether to retain each particular point via independent Bernoulli trial, namely draw $b_\theta \sim \text{Bernoulli}(q(\theta))$, and accept it if $b_\theta = 1$. The Corollary 5.1 implies that the resultant point process remains a Poisson process, with mean measure $q\mu$.

This operation can be utilized for dependent DP construction. Let D be a Dirichlet process over Ω as $D \sim \text{DP}(\mu)$, with the stick breaking representation given by

$$D = \sum_{i=1}^{\infty} \pi_i \delta_{\theta_i}. \tag{5.48}$$

Again, we can get the underlying Poisson process as $\Pi^* = \mathcal{P}^*(D; g) = \{(\theta_i, g\pi_i) : i = 1, 2, \ldots\}$ with $g \sim \text{Gamma}(\mu(\Omega))$, which has $\Pi^* \sim \text{DP}(q\mu \times \gamma)$. Now we apply the subsampling procedure to Π^*, independently drawing $b_i \sim \text{Bernoulli}(q(\theta_i))$ for each i, and accepting only those with $b_i = 1$. This results in a subset given by $\Pi^*_{sub} = \{(\theta_i, g\pi_i) : b_i = 1\}$. Here, Π^*_{sub} corresponds to a Gamma process, as

$$G_{sub} = \sum_{i:b_i=1} g\pi_i \delta_{\theta_i} \sim \Gamma\text{P}(q\mu) \tag{5.49}$$

Normalizing G_{sub} yields a Dirichlet process D_{sub}:

$$D_{sub} = S_q(D) \triangleq \sum_{i:b_i=1} \pi'_i \delta_{\theta_i} \sim \text{DP}(q\mu), \quad \text{with } \pi'_i = \frac{\pi_i}{\sum_{j:b_j=1} \pi_j}. \tag{5.50}$$

Here, we use S_q to denote the subsampling operation applied to a DP with acceptance function q. We can see that the construction of D_{sub} does not actually require explicit instantiation of the underlying Poisson and Gamma processes. This can be done by randomly choosing a subset of the terms in D via independent Bernoulli trial, and re-normalizing the coefficients of the selected terms. For this process, we note:

1. The sub-sampling via independent Bernoulli trial fundamentally differs from choosing a random subset of fixed size. That the acceptance of each term is independently determined is the key in our method to preserve complete randomness.

2. In general, the acceptance function q can be any measurable function with values in $[0, 1]$. This provides the flexibility for one to incorporate application-specific knowledge that favors some subset of Ω. However, in many practical cases, it is sufficient to just use a constant q, which would make the computation easier.

Point Transition

The third way to construct a dependent DP is *point transition*. A point transition operation moves each term independently following a *probabilistic transition function*, which is formally defined below.

Definition 5.4 (Probabilistic transition function)**.** *A probabilistic transition function over a measurable space $(\Omega, \mathcal{F})$ is a function $T : \Omega \times \mathcal{F} \to [0, 1]$ such that*

1. *for each $\theta \in \Omega$, $T(\theta, \cdot)$ is a probability measure over Ω;*
2. *for each $A \in \mathcal{F}$, $p(\cdot, A)$ is integrable.*

The probabilistic transition function describes, in a probabilistic manner, how a point in Ω would move at each time step. In particular, $T(\theta, \cdot)$ is the conditional distribution of where a point will be, given that its current location is θ. With slight abuse of notation, we use T as a transform that can act on a point, a set, or a probability measure. Specifically, given $\theta \in \Omega$, $T(\theta)$ is a random variable representing the destination of θ. Given a set $S \subset \Omega$, $T(S)$ represents a random point process, of which each realization is a set of transformed points, as $T(S) \triangleq \{T(\theta) : \theta \in S\}$. Given a probability measure μ over Ω, $T\mu$ is a transformed measure over Ω, given by

$$(T\mu)(A) \triangleq \int_{\Omega} T(\theta, A)\mu(d\theta). \tag{5.51}$$

Intuitively, if μ is a probability distribution of the initial positions, then $T\mu$ is the distribution of the destinations.

Point transition is an operation on a point process that moves each point according to a given transition function T. According to the following theorem, it also preserves

complete randomness, and thus, when applied to a Poisson process, yields a new Poisson process.

Theorem 5.4 (Transition Theorem of Poisson Processes). *Let* $\Pi \sim \mathrm{PP}(\mu)$ *be a Poisson process over* Ω *and* T *be a probabilistic transition. Then*

$$T(\Pi) \triangleq \{T(\theta) : \theta \in \Pi\} \sim \mathrm{PP}(T\mu). \tag{5.52}$$

Like other complete randomness preserving operations, point transition can be used to construct dependent DPs. Let $D \sim \mathrm{DP}(\mu)$ be a DP over Ω, with stick breaking representation $D = \sum_{i=1}^{\infty} \pi_i \delta_{\theta_i}$, and its underlying Poisson process is $\Pi^* = \{(\theta_i, g\pi_i) : i = 1, 2, \ldots\}$, where $g \sim \mathrm{Gamma}(\mu(\Omega))$. Given a transition function T over Ω, we define a transition T^* over the product space $\Omega \times \mathbb{R}^+$ to be $T^*(\theta, w) \mapsto (T(\theta), w)$. According to Theorem 5.4, applying T^* to Π^* results in a new Poisson process, as

$$T^*(\Pi^*) = \{(T(\theta_i), g\pi_i) : i = 1, 2, \ldots\} \sim \mathrm{PP}(T^*(\mu \times \gamma)). \tag{5.53}$$

In addition, we have

Lemma 5.3. *Let* T *be a probabilistic transition over* Ω, T^* *be a probabilistic transition over* $\Omega \times \mathbb{R}^+$ *given by* $T^*((\theta, w)) \mapsto (T(\theta), w)$. *Then given a measure* μ *over* Ω, *and a measure* ν *over* $\mathbb{R}^+$, *then*

$$T^*(\mu \times \nu) = T\mu \times \nu. \tag{5.54}$$

Combining Eq.(5.53) and Eq.(5.54), we immediately get

$$T^*(\Pi^*) \sim \mathrm{PP}(T\mu \times \gamma). \tag{5.55}$$

Then, using the relations between Poisson and Dirichlet processes, we can derive the corresponding Dirichlet process, denoted by $T(D)$, as

$$T(D) \triangleq \sum_{i=1}^{\infty} \pi_i \delta_{T(\theta_i)} \sim \mathrm{DP}(T\mu). \tag{5.56}$$

This operation randomly moves the location of each term, without changing the coefficients, and therefore, we don't have to re-normalize them.

5.3.4 A Markov Chain of Dirichlet Processes

Integrating these three operations, we develop a Markov chain of Dirichlet processes. In this model, given D_{t-1}, the DP at time $t-1$, the generation of D_t is formulated as

$$D_t = T(S_q(D_{t-1})) \oplus H_t, \quad \text{with } H_t \sim \mathrm{DP}(\nu). \tag{5.57}$$

Starting with D_{t-1}, we first choose a subset of terms according to a given acceptance function q, via independently Bernoulli trial. Then the selected terms will be moved around following a probabilistic transition T, which are then combined with some new terms from H_t to form D_t. Here, we call H_t the *innovative process*, as it brings in new terms that were not existent before. By this construction, *creating new terms*, *removing existing terms*, and *varying term locations* are all allowed during the evolution, respective via *superposition*, *subsampling*, and *point transition*. Let μ_t be the base measure of D_t, then we have

$$\mu_t = T(q\mu_t) + \nu. \tag{5.58}$$

In particular, if the acceptance probability q is a constant over the entire space Ω, then

$$\alpha_t = q\alpha_{t-1} + \alpha_\nu. \tag{5.59}$$

Here, $\alpha_t \triangleq \mu_t(\Omega)$ and $\alpha_\nu \triangleq \nu(\Omega)$ are respectively the concentration parameters of μ_t and ν. According to this relation, one may hold the concentration parameter α_t fixed at α_μ over time, by setting $\alpha_\nu = (1-q)\alpha_\mu$. Intuitively, the value of q controls the prior preference of inheriting existing terms to creating new ones.

In addition, we derive the following result that quantifies the statistical dependencies between DPs at different time steps. Given a measurable subset $A \in \mathcal{F}$, and

$s, t \in \mathbb{R}$ we have

$$\mathrm{Cov}(D_s(A), D_{s+t}(A)) \leq \left(\sup_{\theta \in A} q(\theta)\right)^t \mathrm{Var}(D_t(A)). \tag{5.60}$$

This shows that the covariance between two DPs decays exponentially as the temporal distance between them increases, provided that q is upper bounded by some $q_{sup} < 1$. This is often a desirable property in practice.

In this work, we use this Markov chain of DPs as a prior of dynamic mixture models. This provides a temporal evolution mechanism that inherently supports *creation of new components*, *removal of existing components*, and *variation of component parameters*, while maintaining the complete randomness of the underlying Poisson processes, such that D_t at each time t is marginally a Dirichlet process.

5.4 Gibbs Sampling Algorithm

This section presents the sampling algorithm to perform inference over the dynamic mixture model with the Markov chain of DPs as the prior. The key idea here is to derive the sampling steps by exploiting the fact that our construction maintains the property of being marginally DP. Specifically, in section 5.4.1, we first study the posterior distribution of a DP given a finite set of its samples, and derive a construction of DPs that is equivalent to sampling from this posterior distribution. In section 5.4.2, we then develop a sequential sampling procedure to draw samples from a Markov chain of DPs. In section 5.4.3, we further incorporate the generative model of observations into the framework, and adapt the aforementioned algorithm to sample from the posterior distribution of mixture model parameters.

5.4.1 Posterior and Predictive Distributions

Consider the first two steps of the Markov chain in Eq.(5.57), as

$$\begin{aligned} D_0 &\sim \mathrm{DP}(\mu_0), \\ H &\sim \mathrm{DP}(\nu), \\ D_1 &= T(S_q(D_0)) \oplus H. \end{aligned} \tag{5.61}$$

At time $t = 0$, we can draw samples from D_0 following the Pólya urn scheme that has D_0 marginalized out. Then, given a collection of samples observed at $t = 0$, which we denote by Φ_0, to draw samples at $t = 1$, we have to first marginalize out both D_0 and D_1, as their realizations are infinite series that cannot be completely represented by a computer. In particular, this procedure comprises four steps: (1) derive the posterior distribution of D_0, conditioned on Φ_0; (2) derive the distribution of D_1, conditioned on Φ_0, by marginalizing out D_0; (3) derive the predictive distribution of the first sample at $t = 1$, conditioned on Φ_0, by further marginalizing out D_1; and (4) update the predictive distribution of next sample, as we see more samples at $t = 1$.

The derivation in these steps is nontrivial. However, as we shall see, the theoretical connections between Poisson, Gamma, and Dirichlet processes provide a useful perspective to study these distributions and greatly simplify the derivation.

The Posterior Distribution: $D_0|\Phi_0$

The first question is *what's the posterior distribution of D, given the observed samples therefrom?* To answering this question, we establish the following theorem, which gives a construction of D that is equivalent to sampling D from the posterior distribution.

Theorem 5.5. *Given the generative model as above, and a set of samples Φ, which are independently drawn from D. Suppose Φ comprises K distinct atoms: $\phi_1, \ldots, \phi_K$,*

and ϕ_k appears $m(k)$ times, then we have

$$D_{pos} \triangleq \left(g + \sum_{k=1}^{K} w_k\right)^{-1} \left(gD_r + \sum_{k=1}^{K} w_k \delta_{\phi_k}\right) \sim D|\Phi. \tag{5.62}$$

Here, $g \sim \text{Gamma}(\mu(\Omega))$, $w_k \sim \text{Gamma}(m(k))$ for $k = 1, \ldots, K$ are independent Gamma variables, and $D_r \sim \text{DP}(\mu)$ is an independent DP.

The representation of the posterior samples given in Eq.(5.62) has two parts: the one comprised of observed atoms, and the remaining part that is based on D_r. As more samples are observed, the former part would gradually become dominant, and D_{pos} would recover the underlying D exactly as $|\Phi|$ approaches infinity.

The Distribution: $D_1|\Phi_0$

Before continuing the derivation, we first clarify some notations. Instead of explicitly instantiating every sample, we maintain a set of atoms, and use a label for each sample to associate it with the corresponding atom. Each atom in the collection is bound to an index that uniquely identifies it throughout the entire process. The value of an atom may change over time.

In particular, we use K_t to denote the total number of atoms ever observed up to time t, and ϕ_k^t to denote the value of the k-th atom at time t. At each time step (say t), we draw n_t samples, denoted by $\theta_{t:1}, \ldots, \theta_{t:n_t}$, and their labels are denoted by $z_{t:1}, \ldots, z_{t:n_t}$, such that $\theta_{t:i} = \phi_{z_{t:i}}$. In the actual problem, we only store the atom values and labels, and as such the value of each sample can be easily determined through the references to atoms via the labels. In addition, we use Φ_t to denote the set of samples at time t, and $m_t(k)$ to denote the number of times the atom ϕ_k appears in Φ_t.

Given a set of samples Φ_0 drawn from D_0, which has K_0 distinct atoms: $\phi_1^0, \ldots, \phi_{K_0}^0$, and the atom ϕ_k^0 was seen $m_0(k)$ times. Then according to Theorem 5.5, the posterior

distribution $D_0|\Phi_0$ is given by

$$D_0 = \frac{1}{Z_0}\left(g_0' D_0' + \sum_{k=1}^{K_0} w_k^0 \delta_{\phi_k^0}\right), \quad \text{with } Z_0 = g_0' + \sum_{k=1}^{K_0} w_k^0. \tag{5.63}$$

Here, $D_0' \sim \mathrm{DP}(\mu_0)$, $g_0' \sim \mathrm{Gamma}(\mu_0(\Omega))$, and $w_k^0 \sim \mathrm{Gamma}(m_0(k))$ for $k = 1, \ldots, K'$ are all independent. The Poisson process underlying D_0 is

$$\Pi_0^* = \Pi^{*\prime}{}_0 \cup \{(\phi_k^0, w_k^0)\}_{k=1}^{K_0}, \qquad \text{with } \Pi^{*\prime}{}_0 \sim \mathrm{PP}(\mu_0 \times \gamma). \tag{5.64}$$

Applying the operations in Eq.(5.61) to Π_0^*, which involve subsampling and moving the points in Π_0^*, and as well incorporating those from the underlying Poisson process of H, denoted by $\Pi_{H_1}^*$, we get a new Poisson process that can be written as

$$\begin{aligned}
\Pi_{1|0}^* &= \Pi_{H_1}^* \cup T(S_q(\Pi^{*\prime}{}_0 \cup \{(\phi_k^0, w_k^0) : k = 1, \ldots, K_0\})) \\
&= \Pi_{H_1}^* \cup T(S_q(\Pi^{*\prime}{}_0)) \cup \{(T(\phi_k^0), w_k^0) : b_k^1 = 1\} \\
&= \Pi^{*\prime}{}_{1|0} \cup \{(T(\phi_k^0), w_k^0) : b_k^1 = 1\}, \quad \text{with } \Pi^{*\prime}{}_{1|0} = \Pi_{H_1}^* \cup T(S_q(\Pi^{*\prime}{}_0)).
\end{aligned} \tag{5.65}$$

Here, $b_k^1 \sim \mathrm{Bernoulli}(q(\phi_k^0))$ for $k = 1, \ldots, K_0$ are independently drawn for each observed atom, which indicates whether to retain the k-th atom at time $t = 1$. Besides, by Theorem 5.2, $\Pi^{*\prime}{}_{1|0}$ also a Poisson process, as $\Pi^{*\prime}{}_{1|0} \sim \mathrm{PP}(\nu + q\mu_0)$. From Eq.(5.65), the DP corresponding to $\Pi_{1|0}^*$ is given by

$$D_{1|0} = \frac{1}{Z_{1|0}}\left(g_{1|0}' D_{1|0}' + \sum_{k=1}^{K_0} b_k^1 w_k^0 \delta_{T(\phi_k^0)}\right), \quad \text{with } Z_{1|0} = g_{1|0}' + \sum_{k=1}^{K_0} b_k^1 w_k^0. \tag{5.66}$$

Here, $g_{1|0}' D_{1|0}' \sim \Gamma\mathrm{P}(\mu_1)$ with $\mu_1 = q\mu_0 + \nu$ is an independent Gamma process. Eq.(5.66) provides a construction of D_1 that is equivalent to directly sampling it from the conditional distribution $D_1|\Phi_0$, where D_0 is marginalized out.

The Predictive Distribution of the Samples from D_1

We are going to sequentially sample $\theta_{1:1}, \ldots, \theta_{1:n_1}$ from D_1. Particularly, to draw the first sample $\theta_{1:1}$, with D_1 marginalized out, we have to study the predictive distribution $\theta_{1:1}|\Phi_0$. From Eq.(5.66), which gives a construction of samples from $D_1|\Phi_0$, we derive $\theta_{1:1}|\Phi_0$, as

$$\theta_{1:1}|\Phi_0 \sim \frac{1}{\alpha_1^0}\left(\alpha_{\mu_1} p_{\mu_1} + \sum_{k=1}^{K_0} q_k(\phi_k^0) m_0(k) T(\phi_k^0, \cdot)\right). \tag{5.67}$$

Here, $\alpha_{\mu_1} = \mu_1(\Omega)$ and p_{μ_1} are respectively the concentration parameter and base distribution of μ, and α_1^0 is a constant given by $\alpha_1^0 = \alpha_{\mu_1} + \sum_{k=1}^{K_0} q_k(\phi_k^0) m_0(k)$. It can be seen that $\theta_{1:1}$ can be drawn from either the base distribution p_{μ_1}, or the transition distributions due to the inherited atoms $T(\phi_k^{\tau_k})$. Specifically, we can draw $\theta_{1:1}$ as follows.

1. Draw the label $z_{1:1} \in \{1, \ldots, K_0, K_0 + 1\}$ with

$$\Pr(z_{1:1} = k) = \begin{cases} q(\phi_k^0) m_0(k)/\alpha_1^0 & (k \leq K_0), \\ \alpha_{\mu_1}/\alpha_1^0 & (k = K_0 + 1). \end{cases} \tag{5.68}$$

2. If $z_{1:1} = k \leq K_0$, we draw $\phi_k^1 \sim T(\phi_k^0, \cdot)$; otherwise $k = K_0 + 1$, we draw $\phi_k^1 \sim p_\mu$. In both cases, we have $\theta_{1:1} = \phi_{z_{1:1}}^1$.

Depending on the value of $z_{1:1}$, and conditional distribution of θ_2 can be in different form. Particularly, if $z_{1:1} = 0$, $\theta_{1:1}$ is a new atom from p_{μ_1}. In this case, we have

$$\theta_{1:2}|\theta_{1:1}, u_1 = 0; \Phi_0 \sim \frac{\alpha_{\mu_1}}{\alpha_1^1} p_{\mu_1} + \sum_{k=1}^{K_0} \frac{q(\phi_k^0) m_0(k)}{\alpha_0^1} T(\phi_k^0, \cdot) + \frac{1}{\alpha_1^1} \delta_{\phi_{K_0+1}^1}. \tag{5.69}$$

Here, $\alpha_1^1 = \alpha_1^0 + 1$. If $z_{1:1} = l \in [1, K_0]$, then $\theta_{1:1}$ is an instance of the atom ϕ_l^1, which immediately follows that (1) the term ϕ_l is retained at time $t = 1$, implying $b_l = 1$, and (2) the value of $\phi_l^1 = T(\phi_l)$ is actually seen. As a consequence, the distribution

of $\theta_{1:2}$, conditioned on both Φ_0 and $\theta_{1:1}$, is given by

$$\theta_{1:2}|\theta_{1:1}, u_1 = l \geq 1; \Phi_0 \sim \frac{\alpha_{\mu_1}}{\alpha_1^1} p_{\mu_1} + \sum_{k:k\neq l} \frac{q(\phi_k^0) m_0(k)}{\alpha_1^1} T(\phi_k^0, \cdot) + \frac{m_0(l)+1}{\alpha_1^1} \delta_{\phi_l^1}. \quad (5.70)$$

Here, $\alpha_1^1 = \alpha_1^0 + (1 - q(\phi_l^0))m_0(l) + 1$. Note that after this step, the weight of ϕ_l is increased from $q(\phi_l^0)m_0(l)$ to $m_0(l) + 1$.

Forward along the Chain

Recursively carrying the analysis over to time t along the Markov chain, we derive the following construction that characterizes the conditional distribution of D_t, given the observations up to time $t-1$: $\Phi_0, \ldots, \Phi_{t-1}$. Let b_k^t denote whether the k-th atom is retained at time t, and τ_k denote the last time step when the atom ϕ_k was observed. In addition, we define μ_t via the recursive formula $\mu_t = q\mu_{t-1} + \nu_t$. Then, we have

$$D_{t|t-1} = \frac{1}{Z_{t|t-1}} \left(g'_{t|t-1} D'_{t|t-1} + \sum_{k=1}^{K_{t-1}} b_k^t w_k^{\tau_k} \delta_{T^{(t-\tau_k)}(\phi_k^{\tau_k})} \right), \quad (5.71)$$

with

$$g'_{t|t-1} D'_{t|t-1} \sim \Gamma\mathrm{P}(\mu_t), \qquad \text{and} \qquad Z_{t|t-1} = g'_{t|t-1} + \sum_{k=1}^{K_{t-1}} b_k^t w_k^{\tau_k}.$$

Constructing $D_{t|t-1}$ as above is equivalent to sampling from $D_t|\Phi_0, \ldots, \Phi_{t-1}$. Comparing Eq.(5.66) and Eq.(5.71), we see that they are similar in structure, except for several differences as explained below.

1. $\phi_k^{\tau_k}$ is the value of the k-th atom last time we saw it (at $t = \tau_k$).

2. b_k^t is a Bernoulli variable with $\Pr(b_k^t = 1) = q_k^t$. Here, $q_k^t \triangleq (q(\phi_k^{\tau_k}))^{t-\tau_k}$ is the probability that the k-th atom remains at time t, which decreases exponentially, as $t - \tau_k$, the number of consecutive steps that we do not observe it, increases.

3. $w_k^{\tau_k}$ is a Gamma distributed variable as $w_k^{\tau_k} \sim \mathrm{Gamma}(M_{t-1}(k))$, where $M_{t-1}(k) \triangleq \sum_{s=0}^{t-1} m_s(k)$ is the total number of times that the atom ϕ_k was observed.

Algorithm 2 The sequential algorithm to draw samples from a DDP

Call `Initialize-Collection`
for $i = 1, \ldots, n$ **do**
  Draw $z_{t:i} \in \{1, \ldots, K_{t-1} + K_{new} + 1\}$ with $\Pr(z_{t:i} = k) \propto \omega_k$.
  if $z_{t:i} = k > K_{t-1} + K_{new}$ **then**
    Draw new atom $\phi_k^t \sim p_{\mu_t}$
    Set $K_{new} := K_{new} + 1$
    Initialize $m_t(k) := 0$, $\omega_k := 0$, and $\omega_{k+1} := \alpha_{\mu_t}$
  else if $m_t(k) = 0$ **then**
    Draw $\phi_k^t \sim T^{(t-\tau_k)}(\phi_k^{\tau_k}, \cdot)$
    Reset $\omega_k := M_{t-1}(k)$
  end if
  Increment $m_t(z_{t:i}) := m_t(z_{t:i}) + 1$, and $\omega_k := \omega_k + 1$.
end for

Subroutine: `Initialize-Collection`
Set Φ_t as empty set
Set $K_{new} := 0$
Set $m_t(k) := 0$ and $\omega_k := q_k^t M_{t-1}(k)$ for each $k = 1, \ldots, K_{t-1}$
Set $\omega_{(K_{t-1}+1)} := \alpha_{\mu_t}$

4. $T^{(n)}$ denotes an n-fold transition, *e.g.* $T^{(2)}(\phi) = T(T(\phi))$. Here, we use $T^{(t-\tau_k)}$ for the atom ϕ_k, because it has been subject to $(t - \tau_k)$-fold transition if it remains existent.

From Eq.(5.71), we derive the predictive distribution of the first sample $\theta_1 \sim D_t$, conditioned on $\Phi_0, \ldots, \Phi_{t-1}$, as

$$\theta_1 | \Phi_0, \ldots, \Phi_{t-1} \sim \frac{1}{\alpha_t^0} \left(\alpha_{\mu_t} p_{\mu_t} + \sum_{k=1}^{K_{t-1}} q_k^t M_{t-1}(k) T^{t-\tau_k}(\phi_k^{\tau_k}, \cdot) \right). \tag{5.72}$$

Here, α_t^0 is a normalization constant given by $\alpha_{\mu_t} + \sum_{k=1}^{K_{t-1}} q_k^t M_{t-1}(k)$.

5.4.2 Sampling from a Dependent DP

Based on the analysis above, we develop a sequential sampling algorithm that directly draws the samples from the DP priors with $D_0, D_1, \ldots$ marginalized out, as given in Algorithm 2. The following is a brief explanation of the algorithm.

1. First of all, we have to initialize Φ_t as an empty set, and set K_{new}, the number of new atoms created at time t, to zero.

2. For convenience, we introduce a weight ω_k for each atom, and initially set $\omega_k = q_k^t M_{t-1}(k)$ for $k = 1, \ldots, K_{t-1}$. Additionally, we set $\omega_{(K_{t-1}+1)} = \alpha$, which represents the weight of creating an new atom.

3. We draw a new sample at each step. Specifically, at the i-th step, we draw the label $z_{t:i}$ from a finite set $\{1, \ldots, K_{t-1} + K_{new} + 1\}$, with the probability values proportional to the weights, *i.e.* $\Pr(z_{t:i} = k) \propto \omega_k$.

4. If $z_{t:i} = K_{t-1}+K_{new}+1$, we have to draw a new atom ϕ from the base distribution p_μ, increment K_{new} by 1, and initialize both ω_k and $m_t(k)$ to zeros. (Note that ω_k and $m_t(k)$ will be immediately incremented to one, before drawing the next sample.)

5. If $z_{t:i} = k \leq K_{t-1} + K_{new}$, there are two possible cases: (1) $m_t(k) > 0$, which implies that the atom ϕ_k has been seen in previous samples, and its value ϕ_k^t has been determined; (2) $m_t(k) = 0$, which implies that this atom is inherited from the previous time step $t-1$, and it is the first sample associated with it at time t. In this case, we have to draw $\phi_k^t \sim T(\phi_k^{t-1}, \cdot)$, and accordingly we have to change the weight ω_k from $q_k^t M_{t-1}(k)$ to $M_{t-1}(k)$, as we are assured that this atom is retained.

6. In all cases, we will increment both the counter $m_t(z_{t:i})$ and the weight $\omega_{z_{t:i}}$ by one.

5.4.3 Gibbs Sampling for Inference over Dynamic DPMM

We use the Markov chain of Dirichlet processes as the prior of dynamic mixture models in our framework. The generative process is formulated as follows: for each

time step t,

$$\begin{aligned} D_t &= T(S_q(D_{t-1})) + H_t, \qquad \text{with } H_t \sim \mathrm{DP}(\nu); \\ \theta_{t:i} &\sim D_t, \quad x_{t:i} \sim L(\theta_{t:i}), \qquad \text{for } i = 1, \ldots, n_t. \end{aligned} \tag{5.73}$$

Here, $L(\theta_i)$ is the observation model with parameter θ_i for generating x_i. In this work, we perform the inference over the model parameters $\theta_1, \ldots, \theta_{n_t}$ by sampling them from the posterior distribution. As we will show below, sampling from the posterior distribution is similar to sampling from the prior, except that the probability values are modulated by the likelihood factors. To see this, let's consider a model with a simplified form:

$$D \sim \mathrm{DP}\left(\sum_{k=1}^{K} \alpha_k p_k\right); \quad \theta \sim D; \quad x \sim L(\theta). \tag{5.74}$$

Here, p_k can be either a base distribution or a transition distribution from an inherited atom. With D marginalized out, the posterior distribution of θ conditioned on x is

$$p_{pos}(\theta|x) \propto \sum_{k=1}^{K} \alpha_k p_k(\theta) f(x;\theta) \tag{5.75}$$

Here, $f(x;\theta)$ is the pdf value of x with respect to the observation model with parameter θ. Let

$$\bar{f}(x;p_k) \triangleq \int_\theta f(x;\theta) p_k(\theta) d\theta, \quad \text{and} \quad \tilde{p}_k(\theta|x) \triangleq \frac{p_k(\theta) f(x;\theta)}{\bar{f}_k(x)}. \tag{5.76}$$

With respect to the prior distribution p_k, $\bar{f}(x;p_k)$ is the marginal pdf of x, and $\tilde{p}_k(\theta|x)$ is the posterior probability of θ. As a result, we can rewrite the posterior distribution of θ as

$$p_{pos}(\theta|x) = \sum_{k=1}^{K} \tilde{\alpha}_k|_x \cdot \tilde{p}(\theta|x), \qquad \text{with } \tilde{\alpha}_k|_x = \frac{\alpha_k \bar{f}(x;p_k)}{\sum_{l=1}^{K} \alpha_l \bar{f}(x;p_l)}. \tag{5.77}$$

Hence, given x, we can sample θ from the posterior distribution as follows. We first draw $u \in \{1, \ldots, K\}$ with $\Pr(u = k) = \tilde{\alpha}_k|_x$, and then draw $\theta \sim \tilde{p}(\theta|x)$.

Algorithm 3 The algorithm to sample from the posterior distribution of a DPMM

Call `Initialize-Collection`
for $i = 1, \ldots, n$ **do**
  Draw $z_{t:i} \in \{1, \ldots, K_{t-1} + K_{new} + 1\}$ with $\Pr(z_{t:i} = k) \propto \omega_k f_{i,k}$. ($f_{i,k}$ is given in Eq.(5.78)).
  if $z_{t:i} = k > K_{t-1} + K_{new}$ **then**
    Draw new atom $\phi_k^t \sim \theta | x_{t:i}$, with respect to the prior p_{μ_t}
    Set $K_{new} := K_{new} + 1$
    Initialize $m_t(k) := 0$, $\omega_k := 0$, and $\omega_{k+1} := \alpha_{\mu_t}$
  else if $m_t(k) = 0$ **then**
    Draw $\phi_k^t \sim \theta | x_{t:i}$, with respect to the prior $T^{(t-\tau_k)}(\phi_k^{\tau_k}, \cdot)$
    Reset $\omega_k := M_{t-1}(k)$
  end if
  Increment $m_t(z_{t:i}) := m_t(z_{t:i}) + 1$, and $\omega_k := \omega_k + 1$.
end for

Based on this analysis, we derive an algorithm to sample $\theta_{t:1}, \ldots, \theta_{t:n_t}$ from the posterior distribution by adapting Algorithm 2 (the details are given in Algorithm 3). The main changes consist in two aspects:

1. The sampling of labels $z_{t:i}$ takes the observation likelihood into account. In particular, we have $\Pr(z_{t:i} = k) \propto \omega_k f_{i,k}$, where $f_{i,k}$ is given by

$$f_{i,k} = \begin{cases} f(x_{t:i};\ \phi_k^t) & (k \leq K_{t-1} + K_{new} \text{ and } m_t(k) > 0), \\ \bar{f}(x_{t:i};\ T(\phi_k^{\tau_k}, \cdot)) & (k \leq K_{t-1} + K_{new} \text{ and } m_t(k) = 0), \\ \bar{f}(x_{t:i};\ p_{\mu_t}) & (k = K_{t-1} + K_{new} + 1) \end{cases} \tag{5.78}$$

2. Given $z_{t:i} = k$, we sample the atom ϕ_k^t from the posterior conditioned on $x_{t:i}$, with respect to the corresponding prior (p_μ or $T(\phi_k, \cdot)$), rather than drawing it from the prior directly.

The procedure presented in Algorithm 3 can be inefficient as it draws each new atom ϕ_k^t merely based on the first observation associated with it. Hence, in practice, we only use this sequential procedure for bootstrapping, and then run a modified version, which is a Gibbs sampling scheme that iterates between two steps: *atom update* and *label update*:

1. **Atom update:** We resample each atom ϕ_k conditioned on all the observations with label k, denoted by X_k, with respect to the corresponding prior p_k. Particularly, if $k \leq K_{t-1}$, ϕ_k is an inherited atom, and thus $p_k = T^{(t-\tau_k)}(\phi_k^{\tau_k})$; otherwise, ϕ_k is a new atom that were drawn from p_{μ_t}, and thus $p_k = p_{\mu_t}$.

2. **Label update:** This is similar to the steps in Algorithm 3. Specifically, for each $i = 1, \ldots, n$, except that after bootstrapping, m_t have actually counted all the samples observed at time t. Therefore, we have to first decrement $m_t(z_{t:i})$ by 1, and after $z_{t:i}$ is updated, we increment $m_t(z_{t:i})$ according to its new value.

5.5 Empirical Results

This section presents experimental results on both synthetic and real data. Firsta set of simulations are done on synthetic data to compare the proposed DDP model with dynamic FMM in describing Gaussian clusters that may change over time. Then, the DDP model is applied to two real world tasks, modeling people flows and analyzing the trends of research topics. These experiments demonstrate the model's practical utility.

5.5.1 Simulations on Synthetic Data

The data for simulations were synthesized as follows. We initialized the model with two Gaussian components, and added new components following a temporal Poisson process (one per 20 phases on average). For each component, the life span has a geometric distribution with mean 40, the mean of each Gaussian component evolves independently as a Brownian motion, and the variance is fixed to 1. We performed the simulation for 80 phases, and at each phase, we drew 1000 samples for each active component.

At each phase, we sample for 5000 iterations, discarding the first 2000 for burn-in, and collecting a sample every 100 iterations for performance evaluation. The particles of the last iteration at each phase were incorporated into the model as a

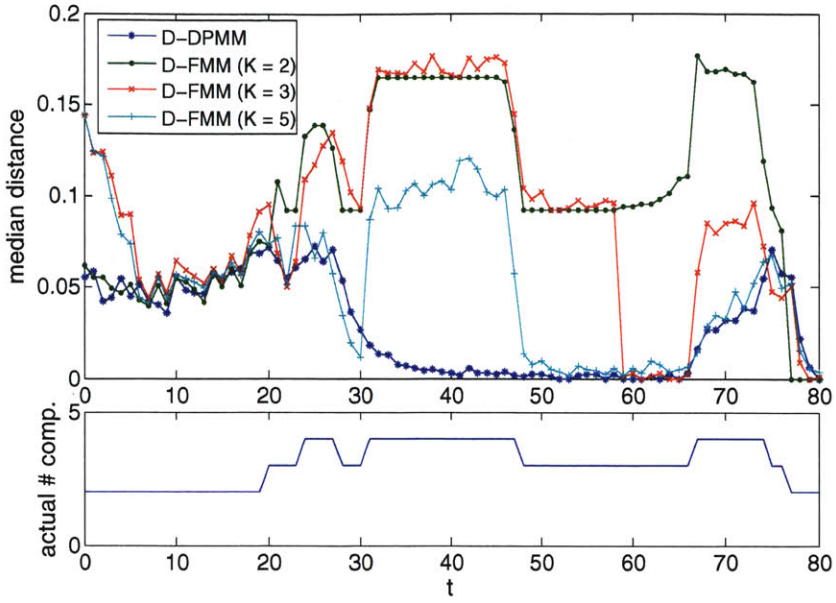
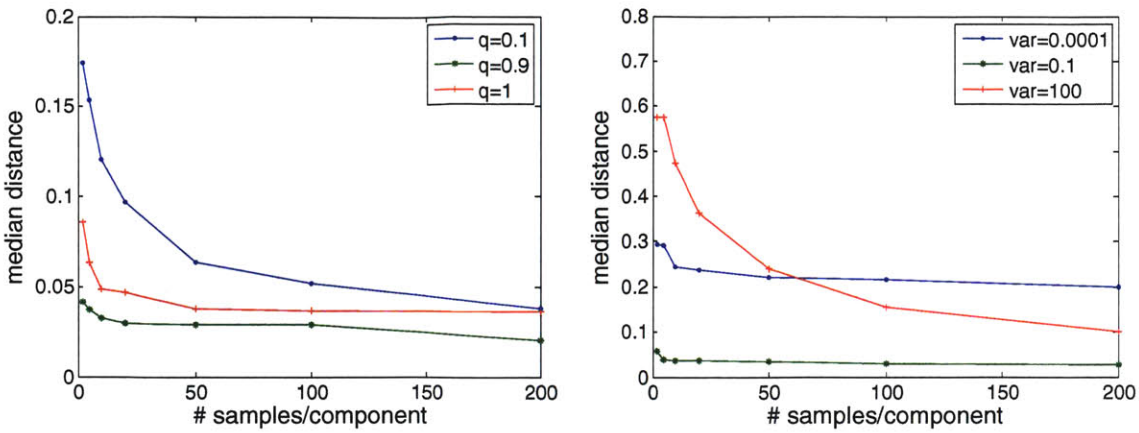


Figure 5-10: This figure compares the performance between D-DPMM and D-FMM with differing numbers of components. The upper graph shows the median of distance between the resulting clusters and the ground truth at each phase. The lower graph shows the actual number of clusters as a function of time. Clearly, the performance of dynamic FMM is inferior to that of dynamic DPMM, when the pre-set number of clusters does not match the true number.

prior for sampling in the next phase. We obtained the label for each observation by majority voting based on the collected samples, and evaluated the performance by measuring the dissimilarity between the resultant clusters and the ground truth using the *variation of information* criterion. Under each parameter setting, we repeated the experiment 20 times, utilizing the median of the dissimilarities for comparison.

We compare our approach (D-DPMM) with dynamic finite mixtures (D-FMM), which assumes a fixed number of Gaussians whose centers vary as Brownian motion. From Figure 5-10, we observe that when the fixed number K of components equals the actual number, they yield comparable performance; while when they are not equal, the errors of D-FMM substantially increase. Particularly, K less than the actual number results in significant underfitting (e.g. D-FMM with $K = 2$ or 3 at phases $30 - 50$ and $66 - 76$); when K is greater than the actual number, samples from the same component are divided into multiple groups and assigned to different components (e.g. D-FMM with $K = 5$ at phases $1 - 10$ and $30 - 50$). In all cases,



(a) For different acceptance prob.

(b) For different diffusion var.

Figure 5-11: The simulation results under different settings: (a) shows the performance of D-DPMM with different values of acceptance probability, under different data sizes. (b) shows the performance of D-DPMM with different values of diffusion variance, under different data sizes.

D-DPMM consistently outperforms D-FMM due to its ability to adjust the number of components to adapt to the change of observations.

We also studied how design parameters impact performance. In Figure 5-11(a), we see that an acceptance probability q to 0.1 creates new components rather than inheriting from previous phases, leading to poor performance when the number of samples is limited. If we set $q = 0.9$, the components in previous phases have a higher survival rate, resulting in more reliable estimation of the component parameters from multiple phases. Figure 5-11(b) shows the effect of the diffusion variance that controls the parameter variation. When it is small, the parameter in the next phase is tied tightly with the previous value; when it is large, the estimation mostly relies on new observations. Both cases lead to performance degradation on small datasets, which indicates that it is important to maintain a balance between inheritance and innovation. Our framework provides the flexibility to attain such a balance. Cross-validation can be used to set these parameters automatically.

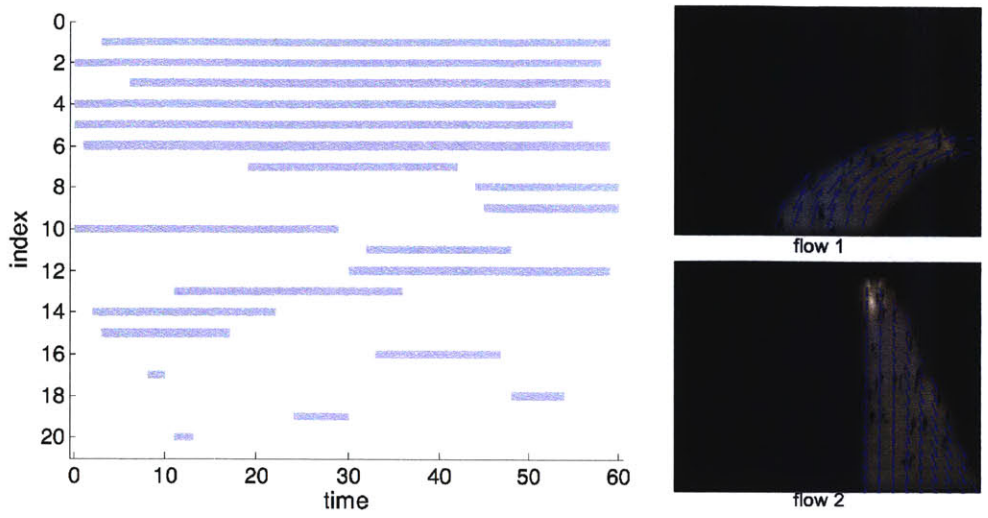


Figure 5-12: The experimental results on people flow modeling. This figure shows the timelines of the top 20 flows. On the right is the snapshot of two such flows, with the velocity fields overlain on the images. (Only the parts covered by the flow domain are visible).

5.5.2 Modeling People Flows

It was observed that the majority of people walking in crowded areas such as a rail station tend to follow motion flows. Typically, there are several flows at a time, and each flow may last for a period. In this experiment, we apply our approach to extract the flows. The test was conducted on video acquired in New York Grand Central Station, which comprises 90,000 frames for one hour (25 fps). A low level tracker was used to obtain the tracks of people, which were then processed by a rule-based filter that discards obviously incorrect tracks. We adopt the geometric flow model described in Chapter 4, which uses an affine field to capture the motion patterns of each flow. The observation for this model is in the form of location-velocity pairs.

We divided the entire sequence into 60 phases (each for one minute), extract location-velocity pairs from all tracks, and randomly choose 3000 pairs for each phase for model inference. The algorithm infers 37 flows in total, while at each phase, the numbers of active flows range from 10 to 18.

Figure 5-12 shows the timelines of the top 20 flows (in terms of the numbers of assigned observations). We compare the performance of our method with D-FMM

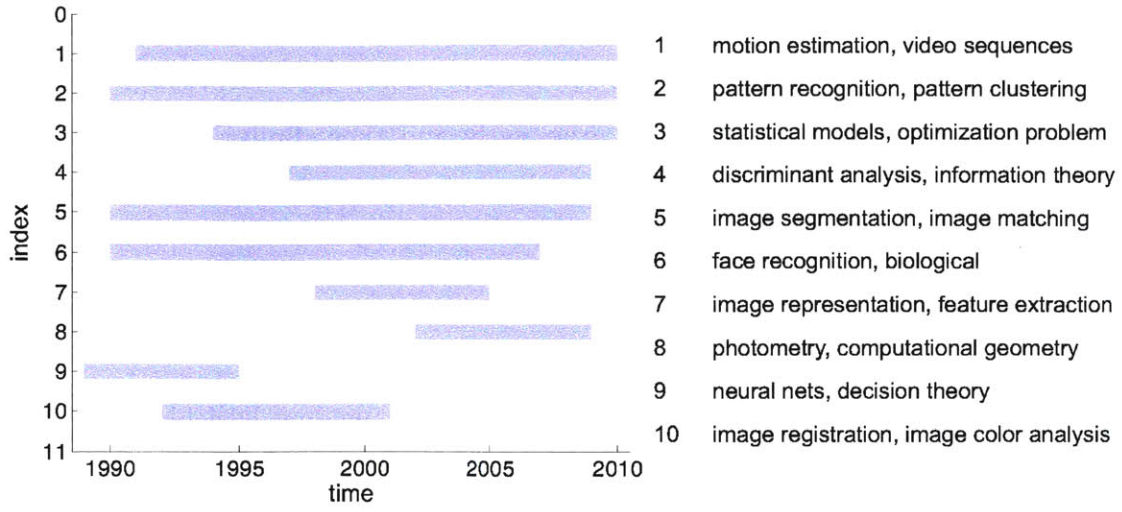


Figure 5-13: The experimental results on PAMI topic analysis. On the left are the timelines of the top 10 topics. On the right are the two leading keywords for these topics.

by measuring the average likelihood on a disjoint dataset. The value for our method is -3.34, while those for D-FMM are -6.71, -5.09, -3.99, -3.49, and -3.34, when K are respectively set to $10, 20, 30, 40$, and 50. Consequently, with a much smaller number of components (12 active components on average), our method attains a similar modeling accuracy as a D-FMM with 50 components.

5.5.3 Analyzing Paper Topics

Next we analyze the evolution of paper topics for IEEE Trans. on PAMI. By parsing the webpage of IEEE Xplore, we collected the index terms for 3014 papers published in PAMI from Jan, 1990 to May, 2010. We divide these papers into 21 groups, each corresponding to a year.

We first compute the similarity between each pair of papers in terms of relative fraction of overlapped index terms. We derive a 12-dimensional feature vector using spectral embedding [74] over the similarity matrix for each paper.

We run our algorithm on these feature vectors (in 21 groups), obtaining a set of clusters. Each cluster can be considered to be related to a particular theme/area.

We compute the histogram of index terms and sorted them in decreasing order

of frequency for each topic. Figure 5-13 shows the timelines of top 10 topics, and together with the top two index terms for each of them. Not surprisingly, we see that topics such as "neural networks" arise early and then diminish while "image segmentation" and "motion estimation" persist.

5.6 Summary

We developed a principled framework for constructing dependent Dirichlet processes. In contrast to most DP-based approaches, our construction is motivated by the intrinsic relation between Dirichlet processes and compound Poisson processes. In particular, we discussed three operations: superposition, subsampling, and point transition. These operations produce DPs depending on others. We further combined these operations to derive a Markov chain of DPs, leading to a prior of mixture models that allows creation, removal, and location variation of component models under a unified formulation. We also presented a Gibbs sampling algorithm for inferring the models. The simulations on synthetic data and the experiments on modeling people flows and paper topics clearly demonstrate that the proposed method is effective in estimating mixture models that evolve over time.

Chapter 6

The Order of Layers

A natural scene that we see in real world typically comprises multiple objects with different appearance and behaviors. As discussed in Chapter 1, these objects can be modeled as layers, and each layer can be described by an appearance model and a motion model. Previous chapters have developed tools to model appearance and dynamic motion. However, this is not enough.

In real scenes, different layers may overlap with each other. Consequently, part of a layer may be occluded by others and become invisible. Effective explanation of such scenes has to take into account the occlusion relation between different layers, which, in turn, is determined by the relative depth order between them.

In this chapter, a model of layer order is developed, which considers the relative depth order between layers as a partial order generated from a distribution over the partial order space. A key challenge, however, arises in inferring the partial order from observations. MAP estimation is NP-hard, while standard MCMC sampling methods are difficult to apply, as the underlying space is generally disconnected, due to the combinatorial constraints that partial orders have to satisfy.

In this chapter, a generic sampling methodology is developed to address this difficulty, which introduces bridging states between parts of the space that are otherwise disconnected. Theoretical analysis of this method leads to bounds of the mixing time. We also tested it empirically in an application to infer the order of layers in real scenes.

6.1 Modeling the Depth Order of Layers

In the layered video model discussed in Chapter 1, a visual scene is considered to be composed of layers, each corresponding to a different object. Each video frame is then generated by combing all the layers according the relative depth order between them (also known as *Z-order*).

6.1.1 Revisiting Layered Video Models

Consider a video with n foreground layers and a background layer. Each layer is associated with a covering domain, an appearance template, and a motion model. At locations covered by multiple overlapping domains, the pixel value is determined by the top layer. Let A_i^t denote the appearance map of the i-th layer at time t, and $A_i^t(x)$ the pixel value at location x. An indicator map L^t maintains the association between pixels and layers, *i.e.* $L^t(x)$ is the index of the top layer at location x. The frame at time t, denoted by I^t, is generated as

$$I^t(x) = A_{L^t(x)}(x) + \varepsilon_t(x). \tag{6.1}$$

Here, $I^t(x)$ is the observed pixel value at x, $\varepsilon_t(x)$ is a noise term, and L^t is determined by the Z-order of the layers, denoted by R^t. An image can be divided into K regions $\omega_1, \ldots, \omega_K$, with each covered by the same set of layers, denoted by S_k. As such, the pixels in ω_k are explained by the same layer, the top one in S_k (*i.e.* $\max_{R^t}(S_k)$). Given R^t and all appearance templates at time t, we have

$$p(I^t|\{A_i^t\}_{i=1}^n; R^t) = \prod_{k=1}^{K} p(I^t(\omega_k)|A_{l^t(\omega_k)}), \quad \text{with } l^t(\omega_k) \triangleq \max_{R^t}(S_k). \tag{6.2}$$

Owing to the lack of order inference techniques, previous methods largely utilize L^t, neglecting the ordering structure. In Wang and Adelson [110], L^t is updated based on local motion similarity. Weiss[113] further incorporated an MRF-prior on L^t to enforce spatial coherence which was followed by a series of related work[57, 114, 89,

41]. Neglecting ordering constraints often leads to an L^t with inconsistent ordering in different parts of the image, particularly when the observations are ambiguous. Jojic and Frey[51] developed a flexible sprite model which explicitly incorporates Z-order, but assumes it is given. Zhou and Tao[118] presented a similar model for object tracking, where the Z-order is inferred via explicit enumeration over all permutations of layers. Such enumeration becomes intractable as the number of layers increases. Sun et al.[96] proposed an alternative by introducing real-valued map with a GP-prior for each layer. L_t is then obtained via thresholding. Importantly, most methods that utilize Z-order assume a total order between layers, however, total order is both unnecessary (one need not consider the ordering of disjoint layers) and inefficient as the number of objects grows. A natural idea is to treat R^t as a partial order, and thereby dispense with unnecessary comparisons.

6.1.2 The Generic Formulation for Partial Order Inference

A general formulation of a probabilistic model over partial orders, suitable for inference, is as follows. Given the partial order R, the conditional likelihood of the observations $\mathbf{x}$ can be written as

$$p(\mathbf{x}|R) = \frac{1}{Z}\prod_{k=1}^{K_{obs}} \phi_k(\max_R(S_k), x_k). \tag{6.3}$$

Here, S_k is a subset of elements, x_k is the part of observations related to this subset, ϕ_k is a potential function describing the appearance within S_k, and Z is a normalization constant. It is not difficult to see that Eq.(6.2) is a special case of this formulation. One can incorporate additional information as a prior on partial orders. For example, in layered video models, it is unlikely that the relative order of two overlapped layers changes across consecutive frames.

Specifically, if layer a is below layer b at time $t-1$, it is likely that this also holds

at t. This prior belief can be formalized as

$$p(R^t|R^{t-1}) = \prod_{\{a_i,b_i\}\in C^{t-1}} p(\max_{R^t}(\{a_i, b_i\})|R^{t-1}), \tag{6.4}$$

where C^{t-1} is a set of comparable pairs for R^{t-1}. Suppose $a_i < b_i$ w.r.t. R^{t-1}, and the chance of switching order is q. Then we have

$$p(\max_{R^t}(\{a_i, b_i\}) = a_i|R^{t-1}) = q. \tag{6.5}$$

In both the prior and the conditional likelihood, the partial order R is incorporated via the maximum of some subsets *w.r.t.* R.

Generally, one can write the joint distribution of the partial order R, the observations $\mathbf{x}$, and other involved parameters $\boldsymbol{\lambda}$, in the following form

$$p(R, \mathbf{x}; \boldsymbol{\lambda}) \propto \psi(\boldsymbol{\lambda}) \prod_{k=1}^{K} \phi_k(\max_R(S_k), x_k; \lambda_k). \tag{6.6}$$

Note here that the maximum element $\max_R(S_k)$ for different sub-regions are implicitly related. We will elaborate on these relations in later sections. The discussion throughout the remaining part of this chapter is based on this generic formulation.

6.2 Minimally Sufficient Partial Orders

As discussed above, partial order plays a prominent role in a layered video model. This section provides a detailed discussion of partial orders. We first review the concept of partial orders and then analyze the identifiability conditions under the generative model formulated above. This leads to a notion of minimal sufficiency and a graph representation that is well defined and easy to manipulate.

6.2.1 Basic Concepts of Partial Orders

First, we review some basic concepts of partial orders.

Definition 6.1 (Partial order). *Let X be a set. A relation $R \subset X \times X$ on X is called a* partial order, *if for each $x, y, z \in X$, it satisfies*

1. *(Reflexivity) $(x, x) \in R$;*
2. *(Antisymmetry) $(x, y) \in R$ and $(y, x) \in R \Rightarrow x = y$;*
3. *(Transitivity) $(x, y) \in R$ and $(y, z) \in R \Rightarrow (x, z) \in R$.*

A set X together with a partial order R, denoted by (X, R), is called a partially ordered set.

We use the following notation to represent relations with respect to the partial order R: $x \leq_R y$ indicates $(x, y) \in R$, $x <_R y$ indicates $(x, y) \in R$ and $x \neq y$, $x \geq_R y$ indicates $y \leq_R x$, and $x >_R y$ indicates $y <_R x$. Two elements $x, y \in \mathcal{X}$ are said to be *incomparable* with respect to R if neither $x \leq_R y$ nor $y \leq_R x$. When the partial order R is clear from the context, and subscript R in these notations can be omitted.

Definition 6.2 (Total order). *A partial order R on X is called a* total order *or a* linear order *if any two elements in X are comparable with respect to R.*

Let (X, R) be a partially ordered set, and $S \subset \mathcal{X}$ be a subset. An element $m \in S$ is called the *maximum* of S if $x \leq_R m$ for each $x \in S$. An element $a \in S$ is called a *maximal* element of S if there exists no $y \in S$ such that $a <_R y$. Similarly, we can define *minimum* and *minimal elements.*

Here, we note several useful facts about maximum and maximal elements:

1. Given a subset of a partially ordered set, there may or may not exist a maximum. However, if a maximum exists, it is unique.
2. Each finite subset of a partially ordered set contains at least one maximal element.
3. If a finite subset S contains only one maximal (say m) with respect to it, then m is the maximum; otherwise, all maximal elements are incomparable to each other, and there exists no maximum of S.

4. Any finite subset of a totally ordered set has a unique maximal element, which is also the maximum of the subset.

Since each partial order R is essentially a set of pairs, we can take intersections between partial orders. And it turns out that intersection of partial orders remains a partial order, as stated by the following proposition:

Proposition 6.1. *Let $\mathcal{R}$ be a collection of partial orders on a set X, then the intersection $\bigcap_{R\in\mathcal{R}} R$ remains a partial order.*

Closures of Relations

Next, we introduce several useful concepts related to *closures*. With the notion of closures, we can derive a partial order from an antisymmetric relation, which is useful in later discussion.

In general, a *closure* of a subset S is often referred to a "minimum" superset of S that possesses some properties. Specifically, we give a formal definition of *reflexive closure*, *transitive closure*, and *reflexive transitive closure*.

Definition 6.3 (Reflexive closure). *Let R be a relation on X, the intersection of all reflexive relations on X that contain R is called the* reflexive closure *of R, which can be written as*

$$\mathrm{CL}_{\mathfrak{R}}(R) \triangleq R \cup \{(x,x) : x \in X\}. \tag{6.7}$$

Definition 6.4 (Transitive closure). *Let R be a relation on X, the intersection of all transitive relations on X that contain R is called the* transitive closure *of R, which can be written as*

$$\mathrm{CL}_{\mathfrak{T}}(R) \triangleq \{(x,y) : \exists x = z_0, \ldots, z_K = y \in X,\ s.t.\ (z_{i-1}, z_i) \in R \text{ for } i = 1, \ldots, K\}. \tag{6.8}$$

Definition 6.5 (Reflexive transitive closure). *Let R be a relation on X, the intersection of all relations that are both reflexive and transitive and contain R is called the*

reflexive transitive closure *of R, which can be written as*

$$\mathrm{CL}_{\mathfrak{RT}}(R) = \mathrm{CL}_{\mathfrak{R}}(\mathrm{CL}_{\mathfrak{T}}(R)) = \mathrm{CL}_{\mathfrak{T}}(\mathrm{CL}_{\mathfrak{R}}(R)). \tag{6.9}$$

The following proposition states that any *antisymmetric relation* R can be made into a partial order by constructing its *reflexive transitive closure.*

Proposition 6.2. *Let R be an antisymmetric relation, then its reflexive transitive closure is a partial order, and it is the intersection of all partial orders containing R.*

If R is finite, we can construct its (reflexive) transitive closure within polynomial time (the simplest way is to use *Floyd-Warshall algorithm*).

6.2.2 Sufficiency, Identifiability, and Minimality

In previous section, we established a generic probabilistic model, in which the joint distribution of the partial order R, observations $\mathbf{x}$, and other involved parameters $\boldsymbol{\lambda}$, is given by

$$p(R; \mathbf{x}, \boldsymbol{\lambda}) \propto \psi(\boldsymbol{\lambda}) \prod_{k=1}^{K} \phi\left(y_k, \max_R(S_k)\right). \tag{6.10}$$

Here, $S_1, \ldots, S_k$ are subsets of the layers, $\max_R(S_k)$ is the maximum of S_k with respect to the partial order R, and ψ captures the prior factors that do not depend on the partial order. Based on this model, we can further discuss the sufficiency and minimality of the partial orders.

Sufficient Partial Order and Consistent Choice of Maximums

To perform evaluation or inference on the model given by Eq.(6.10), the partial order R should contain all necessary pairs such that the maximum of each S_k can be determined. Such a partial order is called a *sufficient partial order*, which is defined formally as follows.

Definition 6.6 (Sufficient partial order)**.** *Let X be a set, and $\mathcal{C} \subset 2^X$ be a collection of subsets of X. A partial order R on X is said to be* sufficient *with respect to $\mathcal{C}$ if*

for each $S \in \mathcal{C}$, the maximum of S exists.

Obviously, any total order on X is *sufficient* with respect to any collection $\mathcal{C}$ of finite subsets. However, the converse is not true in general, meaning that a sufficient partial order is not necessarily a total order. Consider a simple case as follows. Let $X = 1, 2, 3$ and $\mathcal{C} = \{\{1, 2\}, \{1, 3\}\}$, then the partial order $R = ((1, 1), (2, 2), (3, 3), (1, 2), (1, 3))$ is sufficient with respect to $\mathcal{C}$, which is obviously not a total order.

Each sufficient partial order gives rise to an assignment of maximum to each element in $\mathcal{C}$. Such assignments together constitute a *choice function* over $\mathcal{C}$, which is formally defined as below.

Definition 6.7 (Choice function). *Let $\mathcal{C}$ be a collection of subsets of X. A* choice function *on $\mathcal{C}$ is a map $f : \mathcal{C} \to X$ such that for each $S \in \mathcal{C}$, $f(S) \in S$.*

Intuitively, a choice function can be understood as *choosing an element* from each subset S. We note that the well-known *axiom of choice* is based on a choice function. Clearly, a sufficient partial order leads to a choice function that maps each subset S in $\mathcal{C}$ to a maximum of S, which we call the *choice of maximums*.

Identifiability

We can see from Eq.(6.10) that the partial order is incorporated into the model via the induced choice of maximums. Therefore, partial orders that yield the same choice of maximums can not be distinguished by this model, as the joint probability evaluates to the same value for any given $\mathbf{y}$. In this case, we say that R_1 and R_2 are *unidentifiable*. Formally, the *identifiability* is defined as follows.

Definition 6.8 (Identifiability). *Given a joint probabilistic model $p(\boldsymbol{\theta}, \mathbf{y})$ and $\boldsymbol{\theta} \in \Theta$. We say that the set of parameters Θ is* identifiable *under this model, if $p(\mathbf{y}|\boldsymbol{\theta}_1)$ and $p(\mathbf{y}|\boldsymbol{\theta}_2)$ are different distributions for any two distinct elements $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$.*

Definition 6.9 (Factor-wise identifiable model). *Given a probabilistic model $p(\boldsymbol{\theta}, \mathbf{y})$*

with a product form as

$$p(\boldsymbol{\theta}, \mathbf{y}) \propto \psi(\mathbf{y}) \prod_{k=1}^{K} \phi_k \left(y_k, \eta_k(\boldsymbol{\theta}) \right). \tag{6.11}$$

Here, y_k refers to the part of $\mathbf{y}$ that is involved in the k-th factor. Note that the parts of $\mathbf{y}$ involved in different factors may overlap. This model is called factor-wise identifiable, *if $\tilde{p}_k(\cdot|\eta) \neq \tilde{p}_k(\cdot|\eta')$ whenever $\eta \neq \eta'$, for each $k = 1, \ldots, K$. Here, $\tilde{p}_k(\cdot|\eta)$ is defined to be*

$$\tilde{p}_k(y_k|\eta) \triangleq \frac{\psi(\mathbf{y})\phi_k(y_k, \eta)}{\int \psi(\mathbf{y})\phi_k(y_k', \eta) d\mathbf{y}'}. \tag{6.12}$$

The factor-wise identifiability means that for each factor in Eq.(6.11), different values of $\eta_k(\boldsymbol{\theta})$ can be distinguished by the factor ϕ_k itself. Under the assumption of factor-wise identifiability, we showed that a sufficient and necessary condition for a set of partial orders to be identifiable is that they yield different choices of maximums, as stated by the following lemma.

Lemma 6.1. *Given a set X, a collection of subsets $\mathcal{C} = (S_1, \ldots, S_K)$. and a probabilistic model as in Eq.(6.10) with given parameters $\boldsymbol{\lambda}$. Suppose this model is factor-wise identifiable, then a set of partial orders is identifiable under this model if and only if they yield distinct choices of maximums.*

Proof. The "only if" part of the statement is trivial to prove, as one can easily see that when two partial orders yield exactly the same choice of maximums, they are obviously unidentifiable. Since $\boldsymbol{\lambda}$ is given and fixed, we omit it in the following derivation for conciseness.

Next, we show the "if" part. Let R_1 and R_2 be two different partial orders in $\mathcal{R}$. To prove they are identifiable, we have to show that $p(\cdot|R_1)$ and $p(\cdot|R_2)$ are different distributions. Without losing generality, we assume that $p(\cdot|R_1)$ is absolutely continuous with respect to $p(\cdot|R_2)$, otherwise, they have already been different, and

we are done. Then the Kullback-Leibler divergence is

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}(p(\cdot|R_1)||p(\cdot|R_2)) &= \int_{\mathbf{y}} p(\mathbf{x}|R_1) \log \frac{p(\mathbf{x}|R_1)}{p(\mathbf{x}|R_2)} \mu(d\mathbf{x}) \\
&= \sum_{k=1}^{K} \int_{\mathbf{y}} p(\mathbf{x}|R_1) \log \frac{\phi(x_k, \max_{S_k}(R_1))}{\phi(x_k, \max_{S_k}(R_2))} \mu(d\mathbf{x}) \\
&= \sum_{k=1}^{K} \int_{x_k} p(x_k|R_1) \log \frac{\phi(x_k, \max_{S_k}(R_1))}{\phi(x_k, \max_{S_k}(R_2))} \int_{\mathbf{x}_{-k}} p(\mathbf{x}_{-k}|R_1) \mu_{-k}(d\mathbf{x}_{-k}) \mu_k(dx_k) \\
&= \sum_{k=1}^{K} \int_{x_k} p(x_k|R_1) \log \frac{\phi_k(x_k, \max_{S_k}(R_1))}{\phi_k(x_k, \max_{S_k}(R_2))} \mu_k(dx_k). \qquad (6.13)
\end{aligned}$$

Here, μ is the base measure for $\mathbf{y}$, which can be the counting measure for discrete distribution or the Lebesgue measure for continuous distribution. $\mathbf{y}_{-k}$ refers to the part of $\mathbf{y}$ excluding y_k. According to the construction given in Eq.(6.10) and Eq.(6.12), it is not difficult to see that (with the common factors canceled out)

$$\tilde{p}_k(x_k|\max_{S_k}(R)) = p(x_k|R) \propto \phi_k(x_k, \max_{S_k}(R)). \qquad (6.14)$$

Substituting this into the derivation above, we get

$$\mathrm{D}_{\mathrm{KL}}\left(p(\cdot|R_1) \;||\; p(\cdot|R_2)\right) = \sum_{k=1}^{K} \mathrm{D}_{\mathrm{KL}}\left(\tilde{p}_k(\cdot|\max_{R_1}(S_k)) \;||\; \tilde{p}_k(\cdot|\max_{R_2}(S_k))\right). \qquad (6.15)$$

By assumption, R_1 and R_2 yield different choices of maximums, and thus there exists at least one k such that $\max_{S_k}(R_1) \neq \max_{S_k}(R_2)$. Combining this with the factor-wise identifiable assumption, we can conclude that the k-th term in this equation, which measures the divergence between $\tilde{p}_k(\cdot|R_1)$ and $\tilde{p}_k(\cdot|R_2)$, is positive. Since all other terms are non-negative, the divergence between $p(\cdot|R_1)$ and $p(\cdot|R_2)$ is positive, implying that $\mathcal{R}$ is identifiable under this model. □

Minimally Sufficient Partial Order

Based upon the choices of maximums that they induce, we can divide all sufficient partial orders into *equivalent classes*, such that the orders within the same class yield

the same choice of maximums, and thus are unidentifiable under the probabilistic model described above. Each such class has a unique representative which we call *minimally sufficient partial order*, based upon which we develop our representation.

Definition 6.10 (Minimally sufficient partial orders). *Given a set X and a collection $\mathcal{C}$ of subsets. A partial order R is said to be* minimally sufficient *with respect to $\mathcal{C}$ if R is sufficient but any of its proper subset is not.*

Definition 6.11 (Consistent choice of maximums). *Given a set of X and a collection $\mathcal{C}$ of subsets. A choice of maximums over $\mathcal{C}$ is called* consistent *if there exists a partial order that yields this choice.*

Clearly, in order for R to induce a choice f over $\mathcal{C}$, every pair in form of $(a, f(S))$ with $a \in S$ must be included in R, which we call an *essential pair*.

Definition 6.12 (Essential pairs). *Given a set of X, a collection $\mathcal{C}$ of subsets, and a consistent choice of maximums f over $\mathcal{C}$. Each pair (a, b) such that there exists $S \in \mathcal{C}$ with $a \in S$ and $b = f(S)$ is called a* essential pair *with respect to f.*

Proposition 6.3. *Given a set X and a consistent choice of maximums f over $\mathcal{C}$. Then a partial order R yields f as the choice of maximum if and only if all essential pairs for f are contained in R.*

Lemma 6.2. *Given a set X, a collection of subsets $\mathcal{C}$, and a consistent choice of maximum f over $\mathcal{C}$. There exists a unique minimally sufficient partial order that yields f, which is the intersection of all sufficient partial orders that yield f as their choice of maximum.*

For conciseness of following discussion, we denote the intersection of all partial orders that yield f, as described by this lemma, to be R°_f. To prove this lemma, we have to show that R°_f is the *unique* minimally sufficient partial order with f as the choice of maximums.

Proof. First of all, by proposition 6.1, R°_f is a partial order on X. In the following, we have to show that it is *sufficient*, *minimal*, and is the *unique minimal* that yields f.

We have known that the essential pairs for f are contained in every sufficient partial order that yields f as the choice of maximum. As a consequence, all such pairs are contained in their intersection R_f°, and thus R_f° is also a sufficient partial order that yields f as the choice of maximum.

Note that R_f° can be equivalently defined as the intersection of all partial orders comprised of all essential pairs for f. It implies for any of its proper subset, say R', at least one essential pair, say $(a, f(S))$, is not contained in R'. Therefore $f(S)$ is not a maximum of S with respect to R'. Moreover, there is no other element in S can be a maximum. *(This claim can be shown as follows: suppose $b \neq f(S)$ is a maximum of S with respect to R'. Then $f(S) <_{R'} b$, and thus $f(S) <_{R_f^\circ} b$ as $R' \subset R_f^\circ$, which contradicts the assumption that $f(S)$ is a maximum of S with respect to R_f°.)* Hence, any proper subset of R is not sufficient, and R is a minimally sufficient partial order.

Finally, we show the uniqueness. Suppose there is a different minimally sufficient partial order R' that induces f. By the definition of R_f°, we know that R_f° is a proper subset of R', which clearly contradicts the assumption that R' is minimally sufficient. □

Lemma 6.1 and Lemma 6.2 together characterize the key relationship between identifiability and minimal sufficiency, as summarized below.

Given a set X, a collection $\mathcal{C} = (S_1, \ldots, S_K)$ of subsets of X. Under the probabilistic model given in Eq.(6.10) with the assumption that it is factor-wise identifiable, we have:

1. *any partial orders that yield the same choice of maximums are unidentifiable;*
2. *given each consistent choice of maximums over $\mathcal{C}$, there exists a unique minimally sufficient partial order with respect to $\mathcal{C}$ that induces it, which is R_f°;*
3. *the set of all minimally sufficient partial orders are identifiable.*

6.2.3 Representation based on Directed Acyclic Graph

Next, we develop a representation of minimally partial orders that can be implemented efficiently. This is a graph representation, derived by exploiting the intrinsic relations between partial orders and *directed acyclic graphs (DAG).*

Some Graph Theoretical Terminologies

Before introducing the representation, we first review some graph theoretical terminologies that will be used in the discussion.

Definition 6.13 (Directed graph). *A* directed graph *(or* digraph*) is a pair* $G = (V, E)$ *such that* $E \subset V \times V$. *Each element* $v \in V$ *is called a* node *or a* vertex. *Each pair* $(u, v) \in E$ *is called an* directed edge, *of which* u *and* v *are respectively called the source node and the target node.*

Definition 6.14 (Directed path). *Given a directed graph* $G = (V, E)$. *A sequence of nodes* $\mathfrak{p} = (v_0, \ldots, v_l)$ *is called a* directed path *if they are all different (no repeated nodes in the sequence), and* $(v_{i-1}, v_i) \in E$ *for* $i = 1, \ldots, l$. *Here,* l *is called the length of* $\mathfrak{p}$.

Definition 6.15 (Directed cycle). *Given a directed graph* $G = (V, E)$. *A sequence of nodes* $(v_0, \ldots, v_l, v_0)$ *is called a* directed cycle *if* $(v_0, \ldots, v_l)$ *is a directed path, and* $(v_l, v_0) \in E$.

Definition 6.16 (Reachability). *Given a directed graph* $G = (V, E)$. *We say that the node* v *is* reachable *from* u *if* $u = v$ *or there exists a directed path from* u *to* v.

We use $u \xrightarrow{G} v$ to indicate that v is reachable from u with respect to G, and $u \overset{G}{\not\to} v$ otherwise. When the underlying graph G is clear from the context, the notation can be simplified as $u \to v$ and $u \not\to v$.

Definition 6.17 (Reachability relation). *Given a directed path* $G = (V, E)$, *its* reachability relation, *denoted by* $R(G)$, *is defined to be*

$$R(G) \triangleq \{(u, v) : u \to v \text{ with respect to } G\}.$$

In other words, $u \leq_{R(G)} v$ if and only if there exists a directed path from u to v.

Definition 6.18 (Directed acyclic graph). *A directed graph $G = (V, E)$ is called a* directed acyclic graph *(often abbreviated to* DAG*), if there is no directed cycle in G, i.e. given two distinct nodes $u, v \in V$, if v is reachable from u via a directed path, then u is not reachable from v.*

In this document, we only consider directed graphs without self-loops (the edges in form of (u, u)). This assumption is applied implicitly throughout the remaining text. Directed acyclic graph has a close relation with partial order, as stated by the following proposition.

Proposition 6.4. *The reachability relation of a directed acyclic graph $G = (V, E)$ is a partial order defined on the set of nodes V.*

The Graph Representation with Essential Pairs

Owing to the inherent relations between directed acyclic graphs and partial orders, we can use a directed acyclic graph to represent a partial order.

Definition 6.19 (Compatible graph representation). *An acyclic directed graph $G = (X, E)$ is called* compatible *with a partial order R on X, if $R = R(G)$, i.e. R is the reachability relation of G. If this holds, we call G a* compatible representation *of R.*

In general, there can be multiple graph representations that are compatible with a partial order. For example, consider a partial order R over the finite set $V = \{a, b, c\}$, given by $a < b < c$. Then the graph $G_1 = \{V, \{(a, b), (b, c)\}\}$ and the graph $G_2 = \{V, \{(a, b), (a, c), (b, c)\}\}$ are both *compatible* with R.

Proposition 6.5. *Given an acyclic directed graph $G = (V, E)$, the reachability relation $R(G)$ is the intersection of all partial orders that contain every pair $(u, v) \in E$.*

Proof. Let R_{inter} be the intersection of all partial orders that contain E. Here, we are to show $R(G) = R_{inter}$. First, it is obvious that $R(G)$ itself contains E, and hence $R_{inter} \subset R(G)$. Next, we show $R(G) \subset R_{inter}$. It suffices to let R' be an

arbitrary partial order that contains every pair $(u, v) \in E$ and show $R(G) \subset R'$. Given $(s, t) \in R(G)$, there exists $u_0, \ldots, u_n$ such that $u_0 = s$, $u_n = t$ and $(u_{i-1}, u_i) \in E$ for $i = 1, \ldots, n$. Since R' is transitive (as it is a partial order), $(s, t) \in R'$. Therefore, $R(G) \subset R'$. □

In practice, especially in a dynamic context where the partial order can vary over time, maintaining an exact representation is inefficient, as this requires keeping track of every pair of reachable nodes, which is often unnecessary. Here, we use the *graph representation with essential pairs*, which is much more efficient to maintain.

Definition 6.20 (Graph representation with essential pairs)**.** *Given a sufficient partial order R with respect to a collection $\mathcal{C}$ of subsets on X. The* graph representation with essential pairs, *denoted by $G^{ess}(R)$, is defined to be $G^{ess}(R) \triangleq (X, E^{ess}(R))$ with edges connecting the essential pairs, as*

$$E^{ess}(R) = \{(a, \max_R(S)) : a \in S,\ S \in \mathcal{C}\}. \tag{6.16}$$

Again, we use an example to illustrate the essential graph representation. Consider the partial order given by $a < b < c$, and a collection of subsets $\mathcal{C} = \{\{a, b\}, \{a, b, c\}\}$, then the set of essential pairs is $\{(a, b), (a, c), (b, c)\}$. However, when C contains only one set, as $C = \{\{a, b, c\}\}$, then the set of essential pairs becomes $\{(a, c), (b, c)\}$.

The following theorem establishes this representation as a valid representation for minimally sufficient partial orders.

Theorem 6.1. *Given a sufficient partial order R with respect to a collection $\mathcal{C}$ of subsets of X, the graph representation with essential pairs $G^{ess}(R)$ is a compatible representation of the minimally sufficient partial order that yields the same choice of maximum. In particular, if and only if R is a minimally sufficient partial order with respect to $\mathcal{C}$, $G^{ess}(R)$ is compatible with R.*

Proof. First, we note that according to Lemma 6.2, a given consistent choice of maximums f corresponds *uniquely* to a minimally sufficient partial order, which we denote by R_f°. Suppose f is the choice yielded by the given sufficient partial order R. Then

by Proposition 6.3 and Eq.(6.16), we see that $E^{ess}(R)$ is contained in R, and every partial order containing $E^{ess}(R)$ yields the choice f. According to Proposition 6.5, the reachability relation $R(G^{ess}(R))$ is the intersection of all partial orders that contain $E^{ess}(R)$, which is equivalent to the intersection of all partial orders that yield f as the choice of maximums. By Lemma 6.2, such an intersection is precisely the unique minimally sufficient partial order that yields f. Therefore, $G^{ess}(R)$ is compatible with R°_{f}, and as a result, it is compatible with R when R itself is minimally sufficient with respect to $\mathcal{C}$. □

Data Structure and Efficient Implementation

In practice, one can instantiate the graph representation with essential pairs as a graph data structure (e.g. adjacency list) augmented with cross references between the essential edges and the subsets that require them. In what follows, we describe our implementation. Note that this is just one way to implement the representation, which did offer satisfactory efficiency in our experiments. There can be other ways to implement this.

Given a finite set X with $|X| = n$, a partial order R, and collection of subsets $\mathcal{C} = \{S_1, \ldots, S_K\}$. Our data structure to represent the graph with essential pairs $G^{ess}(R) = (X, E^{ess}(R))$ comprises a list of *edge-set-maps*, each for a node $x \in X$. Such a map associates each outgoing edge e starting from x with a set of *enforcing subsets*, denoted by $\mathcal{S}(e)$. Concretely, each essential edge is in form of $(a, \max_R(S))$, for which S is called an *enforcing subset* of e. Note that each edge e can have one or multiple enforcing subsets. For example, if b is the maximum for both subsets S_1 and S_2, and $a \in S_1 \cap S_2$, then both S_1 and S_2 are enforcing subsets of e.

It is not difficult to see that the space complexity of this data structure is $O(n + m + m')$, where $m \triangleq |E^{ess}(R)|$ is the number of edges, and $m' = \sum_e |\mathcal{S}(e)|$ is the sum of the number of enforcing subsets for every edge, which, in turn, is equal to the total number of enumerated essential edges for every subset in $\mathcal{C}$ (e.g. if an essential edge

is enforced by two subsets, it is counted twice here). Hence, we have

$$m \leq m' = \sum_{k=1}^{K} |S_k| \leq Kn. \tag{6.17}$$

It is often the case that $|S_k| \ll n$ for each k, then under such circumstances $m' \ll Kn$.

With this data structure, the following operations can be done efficiently:

1. Traversing all outgoing edges of a node $x \in X$. The time complexity of getting the set of edges given x, and that of visiting each edge in the set are both $O(1)$.

2. Given an edge $e = (a, b)$, getting the reference to $\mathcal{S}(e)$ takes $O(\log(\deg(a)))$ to retrieve $\mathcal{S}(e)$, if the edge-set-map is implemented as a tree-based associative container. If it is implemented as a hash map, then this operation takes $O(1)$ time, generally at the cost of increased memory demand.

3. If $\mathcal{S}(e)$ is implemented as a balanced tree (e.g. red-black tree), it takes $O(\log(|\mathcal{S}(e)|)$ to test whether it contains S_k, add one element to it, or remove an element from it. If $\mathcal{S}(e)$ is implemented as a hash set, each of these operations takes $O(1)$ time.

In practice, we may have to transform from one partial order to another, during Markov chain based sampling, or dynamic transition. In our approach, the transform is accomplished via a series of operations on the choices of maximums subject to consistency constraints, which would, in return, result in the changes of the underlying graph representation. These operations include *(1) Withdrawal of a choice, (2) query of candidates for a choice, (3) making a choice from candidates, (4) committing a choice.* Among these operations, the third one, namely making a choice from candidates, involves a probabilistic inference procedure that we will discuss in next section. The remaining three operations can be done deterministically through operations on the augmented graph structure as introduced above, which are described below. Some of these operations may temporarily render the maximum element for some sets unavailable. We set up an array $\mathbf{c}$ of length K to facilitate these operations.

Algorithm 4 Withdrawal of the choice of $\max(S_k)$

Ensure: $\mathbf{c}(k) \in X$ and $|S_k| > 1$.
  for all a in S_k **do**
    let $e = (a, \max(S_k))$.
    remove S_k from $\mathcal{S}(e)$.
    if $\mathcal{S}(e)$ becomes empty **then**
      remove edge e from the graph.
    end if
  end for
  set $\mathbf{c}(k) = -1$.

Algorithm 5 Query of candidates for the choice of $\max(S_k)$

Ensure: $\mathbf{c}(k) = -1$.
  set $Lst_k = \emptyset$.
  set $h(x) = 0$ for each $x \in X$, which indicates whether some of its descendants are in S_k.
  for all u in S_k **do**
    if u has not been visited **then**
      launch a DFS from u.
      during the DFS traversal, if any child c of a node v has $c \in S_k$ or $h(c) = 1$, then set $h(v) = 1$.
    end if
    add u to Lst if $h(u) = 0$.
  end for
  return Lst_k.

Specifically, $\mathbf{c}(k) = x \in X$ (in our implementation, each element in X corresponds to an integer in $[0, n-1]$) indicates the value x has been chosen as the maximum of S_k, and $\mathbf{c}(k) = -1$ indicates that the choice of maximum for S_k is not committed (or has been withdrawn).

1. **(Withdrawal of the choice of** $\max(S_k)$**):** this operation is to withdraw the choice of maximum for a set S_k, and accordingly reduce the current graph to be minimally sufficient with respect to a collection without S_k. The operation involves removing S_k from $\mathcal{S}(e)$ for each edge that it enforces, and removing those edges that are no longer essential ($\mathcal{S}(e)$ becomes empty). Note that we only apply this operation to the subset S_k with more than one element, otherwise the choice of $\max(S_k)$ can never change and will not affect others. The steps are

Algorithm 6 Committing a choice a as $\max(S_k)$

Ensure: $\mathbf{c}(k) = -1$ and $a \in Lst_k$.
  for all u in S_k **do**
    if $e = (u, a)$ is not present **then**
      add $e = (u, a)$ as a new edge.
      set $\mathcal{S}(e) = \emptyset$.
    end if
    add e S_k to $\mathcal{S}(e)$.
  end for
  set $\mathbf{c}(k) = a$.

given in Algorithm 4. The time complexity is $O(|S_k|)$, in terms of the number of basic map/set operations.

2. **(Query of candidates for the choice of** $\max(S_k)$**):** the goal of this operation is to get a list of possible elements that can be selected as the $\max(S_k)$ given the choices already committed for other sets. The basic idea is to search all descendants of each element in S_k, and an element $x \in S_k$ can be a candidate if none of its descendants are in S_k. Note that if we do the search for each element in S_k independently, it is inefficient, as many of the operations are actually redundant. We can improve the efficiency by coordinating all these steps as one depth-first-search. The steps are given in Algorithm 5. The time complexity is $O(|\mathcal{D}(S_k)|)$, where $\mathcal{D}(S_k)$ is a set comprised of all elements in S_k and their descendants. In many cases, including layered video modeling, $|\mathcal{D}(S_k)|$ is not much greater than $|S_k|$.

3. **(Committing a choice** a **as** $\max(S_k)$**):** the goal of this operation is to formally accept a value a as the choice of maximum of S_k. To enforce the partial order requirements, it is important that a is selected from the candidates from the set obtained through the query described above. Committing the choice a involves adding corresponding essential pairs, as well as the references between these edges and S_k. The time complexity is $O(|S_k|)$, in terms of the number of basic map/set operations.

6.3 A New Approach to Sampling Partial Orders

The section above has established the unique correspondence between consistent choices of maximums and minimally sufficient partial orders. Consequently, the problem of inferring the partial order between layers can be reduced to the inference of the corresponding choice of maximums.

The posterior of the choice of maximums can be written in the following generic form

$$p(x_1, \ldots, x_K) \propto \prod_{k=1}^{K} w_k(x_k), \quad \text{s.t. } (x_1, \ldots, x_K) \text{ is consistent} \tag{6.18}$$

Although the probabilistic model is in a product form, the variables are not independent from each other, as the assignment of their values have to satisfy some combinatorial constraints, such that they together form a consistent choice. Therefore, when some of the variables are fixed, the remaining variables can usually only take values from a restricted subset of their original domains.

For such a problem, seeking an optimal estimate is in general NP-Hard. Due to the complexity, we have to resort to approximate inference techniques, among which, Monte Carlo sampling is a prominent choice. Towards the goal of addressing this difficulty, we consider a more generic problem, that is, to develop a generic method to sample from a constrained combinatorial space.

6.3.1 Review of Sampling Methods

Inference of combinatorial configurations under specific constraints arises as an important problem in numerous areas of artificial intelligence, including structural learning [32, 28], data mining [77], bioinformatics [106], and circuit verification [54].

Current sampling methods fall mainly into three categories: (1) *Direct sampling* enumerates all possible samples and evaluates their probabilities. This is usually intractable for combinatorial problems as the sample space grows exponentially with the problem scale. (2) *Rejection sampling* generate samples without enforcing the constraints and rejects those that violate them. This can be very inefficient since the

chance of obtaining a valid sample can be extremely low through random sampling from the underlying product space. (3) *Markov Chain Monte Carlo* (MCMC) [108] is a popular method for Bayesian inference. The idea is to construct an ergodic Markov chain which has the desired distribution as its equilibrium distribution, thus reducing sampling to Markov simulation.

MCMC relies on an ergodic Markov chain with rapid mixing. Devising such a chain over a constrained combinatorial space can be challenging. Gibbs sampling, where each transition updates a single variable of the sample, is one of the most widely used MCMC methods. However, in combinatorial problems (*e.g.* the graph coloring problem, where the color of each node must differ from that of its neighbors), there often exist strong and deterministic relations between variables. Hence, the set of possible values for a variable can be severely restricted by the value of others. At times, no single variable update is possible without violating the constraints, thus rendering the underlying Markov chain non-ergodic.

The Metropolis-Hastings algorithm allows for customized proposal kernels, providing for more flexible moves that may break local traps or jump between different spaces. Duane *et al.* [27] proposed Hybrid Monte Carlo, which utilizes Hamiltonian dynamics to drive the evolution of the target state, resulting in larger strides across the space. Swendsen and Wang [97] proposed an algorithm for efficient simulation of Ising models, which partitions the MRF into clusters, and assign a new spin value for each one at a iteration. Barbu and Zhu [8] later reformulated it as an M-H algorithm, and extended it to a broader class of posterior segmentation problems. Green [38, 39] developed Reversible Jump MCMC, which performs Bayesian model selection, by sampling from a mixture of model spaces with different dimensions, via trans-dimensional jumps.

Data-driven strategies that exploit the observed data to generate proposals have received increasing attention, and have been used to solve various problems such as image segmentation [103] and Bayesian structure learning [28]. These algorithms are difficult to generalize to other contexts as they are tailored to specific models (*e.g.* model selection and MRF labeling).

In past decade, some methods have been proposed specifically to address the problem of sampling from combinatorial space. Wei *et al.* [111] proposed WalkSAT that seeks solutions to a boolean satisfiability problem (SAT) via random walks interleaved with simulated annealing steps. Kitchen and Kuehlmann [54] extended this approach to solve problems with mixed boolean/integer constraints, under the Metropolis-Hastings formulation. This approach allows constraint violation, and drives the state towards satisfying solutions using an energy function that incurs costs for the constraints being violated. Barrett and Simma [9] proposed an MCMC method that explicitly addresses the disconnected-space issue. The idea is to assign small probability mass to each invalid state, and use occasional random restarts to jump between different regions. Both methods above sample from "smoothed" versions of the target distribution instead of the exact one, mixing slowly when valid solutions are sparse, and increase the probability of falling in an invalid region. Hamze and de Freitas [43] presented a method to sample from a constrained Ising model through self avoiding walks. It is exact and efficient, but restricted to a specific type of problem.

6.3.2 Bridging Markov Chains

Suppose we wish to sample from distribution $\boldsymbol{\mu}$ over a constrained combinatorial space X. Using local moves, we can derive a Markov chain with transition matrix $\mathbf{P}$, which may have slow mixing or even be non-ergodic. In order to mitigate such issues we suggest the notion of "bridging" as a way to connect different regions of the sample space that are otherwise difficult or even impossible to communicate.

Specifically, we introduce a set of "bridging states", denoted by Y. Connecting the states in Y with those in X, we obtain a joint chain over the union space $X \cup Y$. If the joint chain is ergodic and has a stationary distribution in form of $(\alpha\boldsymbol{\mu}_X, (1-\alpha)\boldsymbol{\mu}_Y)$ then sampling from $\boldsymbol{\mu}_X$ is equivalent to drawing samples from $X \cup Y$ via the joint chain and discarding those from Y.

With a goal of constructing a joint chain that is ergodic and mixes rapidly, in this section, we discuss the generic problem of bridging between two arbitrary finite Markov chains over disjoint state spaces such that the stationary distributions over

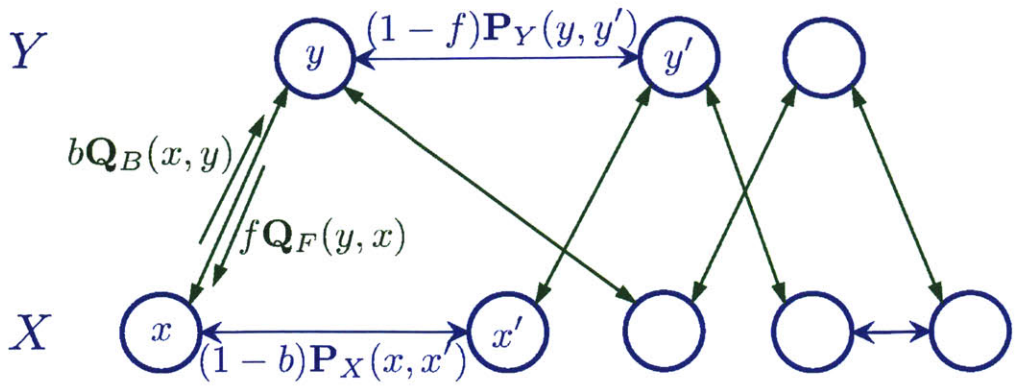


Figure 6-1: This illustrates how two Markov chains are bridged. In the joint chain over $X \cup Y$, each $x \in X$ has a probability $b\mathbf{Q}_B(x,y)$ to transit to $y \in Y$, and each y has a probability $f\mathbf{Q}_F(y,x)$ to transit to x.

the respective spaces are preserved. We also derive bounds of the mixing time, which are influenced by two factors: the *bottleneck ratio* and *laziness.*

Formulation

Consider two finite state spaces X and Y. Suppose we are given two Markov chains: one over X with transition matrix $\mathbf{P}_X$ and stationary distribution $\boldsymbol{\mu}_X$, the other over Y with transition matrix $\mathbf{P}_Y$ and stationary distribution $\boldsymbol{\mu}_Y$. By introducing links that connect between X and Y, as shown in Figure 6-1, we derive the joint transition matrix, as

$$\mathbf{P}_+ = \begin{bmatrix} (1-b)\mathbf{P}_X & b\mathbf{Q}_B \\ f\mathbf{Q}_F & (1-f)\mathbf{P}_Y \end{bmatrix}. \tag{6.19}$$

Here, $\mathbf{Q}_B$ is a $|X| \times |Y|$ matrix, $\mathbf{Q}_F$ is a $|Y| \times |X|$ matrix, and each row in these matrices sums to 1. The behavior of the joint chain is described as follows: Starting from some $x \in X$ samples follow the original chain $\mathbf{P}_X$ with probability $1-b$ and jump to Y with probability b landing at a particular state y with probability $\mathbf{Q}_B(x,y)$. Similarly, starting from $y \in Y$, sampling either stays in Y or jumps to X, respectively with probabilities $1-f$ and f.

While sampling from the joint chain we wish to preserve the stationary distributions $\boldsymbol{\mu}_X$ and $\boldsymbol{\mu}_Y$ within respective spaces, meaning that $\mathbf{P}_+$ has a stationary distribution over $X \cup Y$, in form of $(\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)$ with $\alpha + \beta = 1$. We derive the

following lemma, which establishes the conditions under which this is satisfied.

Lemma 6.3. *The joint transition matrix* $\mathbf{P}_+$ *given by Eq.(6.19) has a stationary distribution in form of* $(\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)$*, if and only if*

$$\boldsymbol{\mu}_X\mathbf{Q}_B = \boldsymbol{\mu}_Y, \quad \textit{and} \quad \boldsymbol{\mu}_Y\mathbf{Q}_F = \boldsymbol{\mu}_X. \tag{6.20}$$

Under this condition, we have $\alpha b = \beta f$*. Further, if both* $\mathbf{P}_X$ *and* $\mathbf{P}_Y$ *are both reversible, then* $\mathbf{P}_+$ *is also reversible, if and only if*

$$\boldsymbol{\mu}_X(x)\mathbf{Q}_B(x,y) = \boldsymbol{\mu}_Y(y)\mathbf{Q}_F(y,x), \tag{6.21}$$

for all $x \in X$ *and* $y \in Y$*.*

Proof. Recall that $\mathbf{P}_+$ is given by

$$\mathbf{P}_+ = \begin{bmatrix} (1-b)\mathbf{P}_X & b\mathbf{Q}_B \\ f\mathbf{Q}_F & (1-f)\mathbf{P}_Y \end{bmatrix}.$$

Suppose $\mathbf{P}_+$ has a stationary distribution in form of $(\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)$, then

$$\begin{aligned} \alpha(1-b)\boldsymbol{\mu}_X\mathbf{P}_X + \beta f\boldsymbol{\mu}_Y\mathbf{Q}_F &= \alpha\boldsymbol{\mu}_X, \\ \alpha b\boldsymbol{\mu}_X\mathbf{Q}_B + \beta(1-f)\boldsymbol{\mu}_Y\mathbf{P}_Y &= \beta\boldsymbol{\mu}_Y. \end{aligned} \tag{6.22}$$

Since $\boldsymbol{\mu}_X$ and $\boldsymbol{\mu}_Y$ are respectively stationary distributions of $\mathbf{P}_X$ and $\mathbf{P}_Y$, *i.e.* $\boldsymbol{\mu}_X\mathbf{P}_X = \boldsymbol{\mu}_X$ and $\boldsymbol{\mu}_Y\mathbf{P}_Y = \boldsymbol{\mu}_Y$, we have

$$\beta f\boldsymbol{\mu}_Y\mathbf{Q}_F = \alpha b\boldsymbol{\mu}_X, \quad \text{and} \quad \alpha b\boldsymbol{\mu}_X\mathbf{Q}_B = \beta f\boldsymbol{\mu}_Y. \tag{6.23}$$

Note that $\mathbf{Q}_B$ and $\mathbf{Q}_F$ were defined with the condition that each of their rows sums to 1, *i.e.* $\mathbf{Q}_F\mathbf{1}_{|X|} = \mathbf{1}_{|Y|}$ and $\mathbf{Q}_B\mathbf{1}_{|Y|} = \mathbf{1}_{|X|}$. Multiplying $\mathbf{1}$ to the right of both hand sides of either equation results in $\alpha b = \beta f$. It immediately follows that

$$\boldsymbol{\mu}_X\mathbf{Q}_B = \boldsymbol{\mu}_Y, \quad \text{and} \quad \boldsymbol{\mu}_Y\mathbf{Q}_F = \boldsymbol{\mu}_X. \tag{6.24}$$

For the other direction, we assume $\mathbf{Q}_B$ and $\mathbf{Q}_F$ satisfy the conditions above, and $\alpha b = \beta f$. Plugging these conditions into the left hand sides of the equations in Eq.(6.22) results in $(\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)\mathbf{P}_+ = (\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)$, which implies that $(\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)$ is a stationary distribution of $\mathbf{P}_+$. The proof of the first part is done.

Next, we show the second part of the lemma, which is about the reversibility. Let $\boldsymbol{\mu}_+ = (\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)$. Under the condition that $\mathbf{P}_X$ and $\mathbf{P}_Y$ are both reversible, we have for each $x, x' \in X$,

$$\begin{aligned} \boldsymbol{\mu}_+(x)\mathbf{P}_+(x,x') &= \alpha(1-b)\boldsymbol{\mu}_X(x)\mathbf{P}_X(x,x'), \\ \boldsymbol{\mu}_+(x')\mathbf{P}_+(x',x) &= \alpha(1-b)\boldsymbol{\mu}_X(x')\mathbf{P}_X(x',x), \end{aligned} \tag{6.25}$$

and thus $\boldsymbol{\mu}_+(x)\mathbf{P}_+(x,x') = \boldsymbol{\mu}_+(x')\mathbf{P}_+(x',x)$ (due to the reversibility of $\mathbf{P}_X$). Likewise, we can get $\boldsymbol{\mu}_+\mathbf{P}_+(y,y') = \boldsymbol{\mu}_+(y')\mathbf{P}(y',y)$. Hence, $\mathbf{P}_+$ is reversible if and only if $\boldsymbol{\mu}_+(x)\mathbf{P}_+(x,y) = \boldsymbol{\mu}_+(y)\mathbf{P}_+(y,x)$ for each $x \in X$ and $y \in Y$. This can be expanded as

$$\alpha b\boldsymbol{\mu}_X(x)\mathbf{Q}_B(x,y) = \beta f\boldsymbol{\mu}_Y(y)\mathbf{Q}_F(y,x), \tag{6.26}$$

which holds if and only if $\boldsymbol{\mu}_X(x)\mathbf{Q}_B(x,y) = \boldsymbol{\mu}_Y(y)\mathbf{Q}_F(y,x)$ (under the condition $\alpha b = \beta f$). The proof is completed. □

We name the condition of Eq.(6.21) as *cross-space detailed balance*. With this construction, the *total probability of cross-space transition* is given by

$$\eta(b,f) \triangleq \alpha b + \beta f = 2\alpha b = 2\beta f = \frac{2bf}{b+f}. \tag{6.27}$$

The value of $\eta(b,f)$ reflects how frequently X and Y communicate with each other, which, as we shall see, is closely related to the mixing time of the joint chain.

We note that the matrix $\mathbf{Q}_{BF} \triangleq \mathbf{Q}_B\mathbf{Q}_F$ is a stochastic matrix, which actually represents a Markov chain over X, where each transition is via an intermediate state in Y. In particular, to complete a transition starting from x, it transits to $y \in Y$ with probability $\mathbf{Q}_B(x,y)$ and then back to x' with probability $\mathbf{Q}_F(y,x')$. Hence, the probability from x to x' is $\sum_{y\in Y}\mathbf{Q}_B(x,y)\mathbf{Q}_F(y,x') = \mathbf{Q}_{BF}(x,x')$. We call this chain

the *collapsed chain* of $\mathbf{P}_+$ over X. We can similarly get a *collapsed chain* over Y, with transition matrix $\mathbf{Q}_{FB} \triangleq \mathbf{Q}_F\mathbf{Q}_B$.

Intuitively, these chains utilize states in the other space to provide alternative transition routes, which, as stated by the following lemma, also lead to the same stationary distributions.

Lemma 6.4. *If the condition given by Eq.(6.20) holds, then $\boldsymbol{\mu}_X$ and $\boldsymbol{\mu}_Y$ are respectively stationary distributions of $\mathbf{Q}_{BF}$ and $\mathbf{Q}_{FB}$. Moreover, if $\mathbf{P}_+$ is reversible, then both $\mathbf{Q}_{BF}$ and $\mathbf{Q}_{FB}$ are reversible.*

Proof. If $(\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)$ is a stationary distribution of $\mathbf{P}_+$, then Eq.(6.24) holds. Thus,

$$\boldsymbol{\mu}_X\mathbf{Q}_B\mathbf{Q}_F = \boldsymbol{\mu}_Y\mathbf{Q}_F = \boldsymbol{\mu}_X, \tag{6.28}$$

implying that $\boldsymbol{\mu}_X$ is a stationary distribution of $\mathbf{Q}_B\mathbf{Q}_F$. Similarly, we can show that $\boldsymbol{\mu}_Y$ is a stationary distribution of $\mathbf{Q}_F\mathbf{Q}_B$.

Furthermore, if $\mathbf{P}_+$ is reversible, according to Lemma 6.3, we have $\boldsymbol{\mu}_X(x)\mathbf{Q}_B(x,y) = \boldsymbol{\mu}_Y(y)\mathbf{Q}_F(y,x)$ for each $x \in X$ and $y \in Y$. Then for any $x, x' \in X$,

$$\begin{aligned}
\boldsymbol{\mu}_X(x)\mathbf{Q}_{BF}(x,x') &= \boldsymbol{\mu}_X(x)\sum_{y\in Y}\mathbf{Q}_B(x,y)\mathbf{Q}_F(y,x') \\
&= \sum_{y\in Y}\boldsymbol{\mu}_X(x)\mathbf{Q}_B(x,y)\mathbf{Q}_F(y,x') \\
&= \sum_{y\in Y}\boldsymbol{\mu}_Y(y)\mathbf{Q}_F(y,x)\mathbf{Q}_F(y,x').
\end{aligned}$$

Similarly, we can get

$$\boldsymbol{\mu}_X(x')\mathbf{Q}_{BF}(x',x) = \sum_{y\in Y}\boldsymbol{\mu}_Y(y)\mathbf{Q}_F(y,x')\mathbf{Q}_F(y,x).$$

Hence,

$$\boldsymbol{\mu}_X(x)\mathbf{Q}_{BF}(x,x') = \boldsymbol{\mu}_X(x')\mathbf{Q}_{BF}(x',x). \tag{6.29}$$

This implies that $\mathbf{Q}_{BF}$ is reversible. Likewise, we can show that $\mathbf{Q}_{FB}$ is reversible

under the condition that $\mathbf{P}_+$ is reversible. The proof is completed. □

On the other hand, as we will discuss later, the ergodicity and the mixing time of the joint chain also depend on the characteristics of $\mathbf{Q}_{BF}$ and $\mathbf{Q}_{FB}$.

6.3.3 Mixing Time Analysis

The efficiency of a Markov chain is often measured by the *mixing time*. Given an ergodic Markov chain over X, with equilibrium distribution $\boldsymbol{\mu}$, the *mixing time* is

$$t_{mix}(\varepsilon) \triangleq \min\{t : \max_{x \in X} \|\mathbf{P}^t(x, \cdot) - \boldsymbol{\mu}\|_{TV} < \varepsilon\}. \tag{6.30}$$

We assume that the eigenvalues of $\mathbf{P}$ are $1 = \lambda_1 \geq \cdots \geq \lambda_n \geq -1$. Then, the *absolute spectral gap* of $\mathbf{P}$ is defined to be $\gamma_*(\mathbf{P}) \triangleq \min\{1 - \lambda_2, 1 + \lambda_n\}$. The theorem [62] below shows that the mixing time closely relates to this absolute spectral gap.

Theorem 6.2. *Given a reversible Markov chain with transition matrix* $\mathbf{P}$, *and* $\varepsilon \in (0, 1/2)$, *then*

$$\log(1/(2\varepsilon))(\tau - 1) \leq t_{mix}(\varepsilon) \leq \log(1/(\varepsilon \boldsymbol{\mu}_{min}))\tau. \tag{6.31}$$

Here, τ *is called the relaxation time, given by* $1/\gamma_*(\mathbf{P})$.

In general, a chain tends to have slow mixing when the absolute spectral gap is small, and when the gap is zero, the chain is non-ergodic and never mixes. The absolute spectral gap depends on two factors, namely the *bottleneck ratio*, which affects the value of $1 - \lambda_2$, *i.e.* the *spectral gap*, and the *laziness of transition*, which influences $1 + \lambda_n$.

Flows and Bottleneck Ratio

Given a Markov chain with transition matrix $\mathbf{P}$, which has a stationary distribution $\boldsymbol{\mu}$. For $x, x' \in X$, we define the *transition flow* (or simply *flow*) from x to x' to be $\mathcal{F}(x, x') \triangleq \boldsymbol{\mu}(x)\mathbf{P}(x, x')$. For a reversible chain, the flows are symmetric, *i.e.* $\mathcal{F}(x, x') = \mathcal{F}(x', x)$. The notion of flow can also be extended to sets. Let A

and B be subsets of X, then the flow from A to B is defined to be $\mathcal{F}(A,B) \triangleq \sum_{x\in A}\sum_{x'\in B}\mathcal{F}(x,x')$.

Consider a partition of X into two subsets S and its complement S^c, then the *transition flow ratio* of S is $\Phi(S,S^c;\mathbf{P}) \triangleq \mathcal{F}(S,S^c)/\min\{\boldsymbol{\mu}(S),\boldsymbol{\mu}(S^c)\}$, where $\boldsymbol{\mu}$ is used as a measure, *i.e.* $\boldsymbol{\mu}(S)=\sum_{x\in S}\boldsymbol{\mu}(x)$. Taking the minimum of such ratio values of all partitions, we get the *bottleneck ratio*, which is formally defined as

$$\Phi_*(\mathbf{P}) = \min_{S\subset X}\left\{\frac{\mathcal{F}(S,S^c)}{\min\{\boldsymbol{\mu}(S),\boldsymbol{\mu}(S^c)\}} : S, S^c \neq \emptyset\right\}. \tag{6.32}$$

Jerrum and Sinclair [50] derived the theorem below that establishes both a lower and upper bound of the spectral gap in terms of bottleneck ratio.

Theorem 6.3. *Let λ_2 be the second largest eigenvalue of a reversible transition matrix $\mathbf{P}$, then*

$$\Phi_*^2(\mathbf{P})/2 \leq 1-\lambda_2 \leq 2\Phi_*(\mathbf{P}). \tag{6.33}$$

This theorem shows that increasing the bottleneck ratio tends to expand the spectral gap, and thus reduce the mixing time. Through theoretical study, we found that the bottleneck ratio of the joint chain $\mathbf{P}_+$ given by Eq.(6.19) depends on both *how frequently the chain jumps between X and Y* and *how well the forward and backward links couple with each other*. The former is controlled by f and b, while the latter is mainly reflected by the spectral structure of the coupled chain: $\mathbf{Q}_{BF}$ and $\mathbf{Q}_{FB}$. We further derived specific bounds that characterize their relations:

Theorem 6.4. *The reversible transition matrix $\mathbf{P}_+$ as given by Eq.(6.19) has*

$$\frac{\eta(b,f)}{2}\cdot\frac{\phi}{\phi+1} \leq \Phi_*(\mathbf{P}_+) \leq \max\{b,f\}. \tag{6.34}$$

Here, $\eta(b,f) = 2\alpha b = 2\beta f$ is the total probability of cross-space transition, $\phi = \min\{\Phi_*(\mathbf{Q}_{BF}),\Phi_*(\mathbf{Q}_{FB})\}$.

This theorem gives both a lower bound and an upper bound of the bottleneck ratio of $\mathbf{P}_+$. We can see that the bottleneck ratio is influenced by two factors: (1) *the*

frequency of cross-space transition. Frequent transition between X and Y generally results in high bottleneck ratio; while if the communication between them is inactive, the bottleneck ratio would be very low, leading to slow mixing. (2) *the bottleneck ratio of the collapsed chains.* High bottleneck ratios of the collapsed chains indicate that transition between different regions is made easy with the intermediate states, and thus the joint chain can mix rapidly. More importantly, this theorem leads to:

Corollary 6.1. *The joint chain* $\mathbf{P}_+$ *is ergodic when the collapsed chains (*$\mathbf{Q}_{BF}$ *and* $\mathbf{Q}_{FB}$*) are both ergodic.*

Proof of Theorem 6.4

To prove theorem 6.4, we first establish a lemma on flow decomposition, and then accomplish the proof based on the lemma.

For the joint chain $\mathbf{P}_+$, we analyze its bottleneck ratio by decomposing the flows. Consider a partition of the union space $X \cup Y$ into two parts: $A \cup B$ (with $A \subset X$ and $B \subset Y$) and $A^c \cup B^c$ (with $A^c = X/A$ and $B^c = Y/B$). The flow between them comprises three parts:

$$\mathcal{F}(A, A^c) + \mathcal{F}(B, B^c) + (\mathcal{F}(A, B^c) + \mathcal{F}(B, A^c)).$$

Here, $\mathcal{F}(A, A^c)$ is *the flow within* X, $\mathcal{F}(B, B^c)$ is *the flow within* Y, and $\mathcal{F}(A, B^c) + \mathcal{F}(B, A^c)$ is *the flow between* X *and* Y. The first two are inherited from the original Markov chains. We focus on the third one, which reflects the effect of bridging. For this part of flow, we derive the following lemma by decomposing it along multiple paths.

Lemma 6.5. *Given arbitrary partition of* $X \cup Y$ *into* $A \cup B$ *and* $A^c \cup B^c$ *as described above, we have*

$$\mathcal{F}(A, B^c) + \mathcal{F}(B, A^c) \geq \alpha b \cdot \Phi_*(\mathbf{Q}_{BF})\boldsymbol{\mu}_X(A), \tag{6.35}$$

when $\boldsymbol{\mu}_X(A) \leq \boldsymbol{\mu}_X(A^c)$, *and*

$$\mathcal{F}(A, B^c) + \mathcal{F}(B, A^c) \geq \beta f \cdot \Phi_*(\mathbf{Q}_{FB})\boldsymbol{\mu}_Y(B), \tag{6.36}$$

when $\boldsymbol{\mu}_Y(B) \leq \boldsymbol{\mu}_Y(B^c)$.

Proof. To analyze the flow $\mathcal{F}(A, B^c) + \mathcal{F}(B, A^c)$, we further decompose it along multiple paths. Then, we have

$$\begin{aligned} \mathcal{F}(A, B^c) &= \sum_{x \in A} \sum_{y \in B^c} \alpha b \boldsymbol{\mu}_X(x) \mathbf{Q}_B(x, y) \\ &= \alpha b \sum_{x \in A} \sum_{y \in B^c} \boldsymbol{\mu}_X(x) \mathbf{Q}_B(x, y) \sum_{x' \in X} \mathbf{Q}_F(y, x') \end{aligned} \tag{6.37}$$

In this way, we decompose the flow into a sum of the terms in form of $\boldsymbol{\mu}_X(x)\mathbf{Q}_B(x, y)\mathbf{Q}_F(y, x')$, which we call the *path weight* along $x \to y \to x'$, denoted by $\omega(x, y, x')$. We can then rewrite $\mathcal{F}(A, B)$ as

$$\mathcal{F}(A, B) = \alpha b \sum_{x \in A} \sum_{y \in B} \sum_{x' \in X} \omega(x, y, x'). \tag{6.38}$$

For conciseness, we use $\omega(A, B, C)$ to denote the sum of paths traveling from A, via B, and ending up in C, *i.e.* $\sum_{x \in A} \sum_{y \in B} \sum_{x' \in C} \omega(x, y, x')$. Then, we have

$$\mathcal{F}(A, B^c) = \alpha b(\omega(A, B^c, A) + \omega(A, B^c, A^c)), \tag{6.39}$$

$$\mathcal{F}(A^c, B) = \alpha b(\omega(A^c, B, A) + \omega(A^c, B, A^c)). \tag{6.40}$$

As $\mathcal{F}$ is symmetric for a reversible chain, we have $\mathcal{F}(B, A^c) = \mathcal{F}(A^c, B)$, and thus

$$\begin{aligned} \mathcal{F}(A, B^c) + \mathcal{F}(B, A^c) &\geq \alpha b(\omega(A, B^c, A^c) + \omega(A^c, B, A)) \\ &= \alpha b\, \omega(A, Y, A^c). \end{aligned} \tag{6.41}$$

On the other hand, we note that

$$\begin{aligned}\omega(A, Y, A^c) &= \sum_{x\in A}\sum_{y\in Y}\sum_{x'\in A^c} \omega(x, y, x') \\ &= \sum_{x\in A}\sum_{y\in Y}\sum_{x'\in A^c} \boldsymbol{\mu}_X(x)\mathbf{Q}_B(x, y)\mathbf{Q}_F(y, x') \\ &= \sum_{x\in A}\sum_{x'\in A^c} \boldsymbol{\mu}_X(x)\sum_{y\in Y}\mathbf{Q}_B(x, y)\mathbf{Q}_F(y, x') \\ &= \sum_{x\in A}\sum_{x'\in A^c} \boldsymbol{\mu}_X(x)\mathbf{Q}_{BF}(x, x'). \end{aligned} \tag{6.42}$$

This is exactly the flow from A to A^c with respect to the collapsed chain $\mathbf{Q}_{BF}$, *i.e.*

$$\omega(A, Y, A^c) = \mathcal{F}_{\mathbf{Q}_{BF}}(A, A^c). \tag{6.43}$$

Assuming $\boldsymbol{\mu}_X(A) \le \boldsymbol{\mu}_X(A^c)$, we have $\mathcal{F}_{\mathbf{Q}_{BF}}(A, A^c) \ge \Phi_*(\mathbf{Q}_{BF})\boldsymbol{\mu}(A)$, by the definition of bottleneck ratio. Combining this with Eq.(6.41) results in

$$\begin{aligned}\mathcal{F}(A, B^c) + \mathcal{F}(B, A^c) &\ge \alpha b \cdot \omega(A, Y, A^c) \\ &\ge \alpha b \cdot \Phi_*(\mathbf{Q}_{BF})\boldsymbol{\mu}_X(A). \end{aligned} \tag{6.44}$$

Likewise, with the assumption $\boldsymbol{\mu}_Y(B) \le \boldsymbol{\mu}_Y(B^c)$, we have

$$\mathcal{F}(A, B^c) + \mathcal{F}(B, A^c) \ge \beta f \cdot \Phi_*(\mathbf{Q}_{FB})\boldsymbol{\mu}_Y(B). \tag{6.45}$$

The proof of the lemma is completed. □

Next, we continue to prove the main theorem.

Proof. Let $\boldsymbol{\mu}_+ = (\alpha\boldsymbol{\mu}_X, \beta\boldsymbol{\mu}_Y)$ be the stationary distribution of $\mathbf{P}_+$. For conciseness, we let $F_s(A, B) \triangleq \mathcal{F}(A \cup B, A^c \cup B^c)$. When A and B are clear from the context, we simply write F_s. Then the bottleneck ratio of $\mathbf{P}_+$ is the minimum of the values of $F_s(A, B)/\boldsymbol{\mu}_+(A \cup B)$, among all possible choices of $A \subset X$ and $B \subset Y$ such that $\boldsymbol{\mu}_+(A \cup B) \le 1/2$ and $A \cup B \ne \emptyset$. Throughout this proof, we assume $A \subset X$, $B \subset Y$,

and $\boldsymbol{\mu}_+(A \cup B) \leq 1/2$, *i.e.* $\alpha\boldsymbol{\mu}_X(A) + \beta\boldsymbol{\mu}_Y(B) \leq 1/2$.

Under this assumption, there are three cases, which we respectively discuss as follows.

Case 1. $\boldsymbol{\mu}_X(A) \leq 1/2$ and $\boldsymbol{\mu}_Y(B) \leq 1/2$.

We have $\phi = \min\{\Phi_*(\mathbf{Q}_{BF}), \Phi_*(\mathbf{Q}_{FB})\}$ in the theorem. In addition, $F_s \geq \mathcal{F}(A, B^c) + \mathcal{F}(B, A^c)$. Combining this with Lemma 6.5, we get

$$F_s \geq \alpha b \cdot \phi\boldsymbol{\mu}_X(A), \text{ and } F_s \geq \beta f \cdot \phi\boldsymbol{\mu}_Y(B). \tag{6.46}$$

Note that $\eta/2 = \alpha b = \beta f$ and $\alpha + \beta = 1$. Thus

$$\frac{F_s}{\boldsymbol{\mu}_+(A \cup B)} \geq \frac{\alpha F_s + \beta F_s}{\alpha\boldsymbol{\mu}_X(A) + \beta\boldsymbol{\mu}_Y(B)} \geq \eta\phi/2. \tag{6.47}$$

Case 2. $\boldsymbol{\mu}_X(A) < 1/2$ and $\boldsymbol{\mu}_Y(B) > 1/2$.

Given arbitrary $\kappa > 2$, there are two possibilities:

Case 2.1. $1/\kappa \leq \boldsymbol{\mu}_X(X) < 1/2$ and $\boldsymbol{\mu}_Y(B) > 1/2$. Then

$$F_s \geq \alpha b \cdot \phi\boldsymbol{\mu}_X(A) > \frac{1}{\kappa}\alpha b\phi. \tag{6.48}$$

Recall that $\boldsymbol{\mu}_+(A \cup B) \leq 1/2$. Thus

$$\frac{F_s}{\boldsymbol{\mu}_+(A \cup B)} \geq \frac{2}{\kappa}\frac{\eta\phi}{2}. \tag{6.49}$$

Case 2.2. $\boldsymbol{\mu}_X(X) < 1/\kappa$ and $\boldsymbol{\mu}_Y(B) > 1/2$. Here, we utilize the following fact: $F_s \geq \mathcal{F}(B, A^c) = \mathcal{F}(B, X) - \mathcal{F}(B, A)$. Then, by the definition of flow, we have

$$\mathcal{F}(B, X) = \beta f\boldsymbol{\mu}_Y(B) > \beta f/2, \tag{6.50}$$

and by the symmetry of $\mathcal{F}$ (due to reversibility),

$$\mathcal{F}(B, A) = \mathcal{F}(A, B) \leq \mathcal{F}(A, Y) = \alpha b/\kappa. \tag{6.51}$$

With $\alpha b = \beta f$, combining the results above leads to

$$F_s > \beta f/2 - \alpha b/\kappa = \alpha b(1/2 - 1/\kappa). \tag{6.52}$$

As a result, we get

$$\frac{F_s}{\boldsymbol{\mu}_+(A \cup B)} > 2\alpha b(1/2 - 1/\kappa) = \frac{\eta}{2}(1 - 2/\kappa). \tag{6.53}$$

Case 3. $\boldsymbol{\mu}_X(A) > 1/2$ and $\boldsymbol{\mu}_Y(B) < 1/2$. Following a similar argument as we developed above for case 2, given $\kappa > 2$, we can likewise get

$$\frac{F_s}{\boldsymbol{\mu}_+(A \cup B)} \geq \begin{cases} \frac{2}{\kappa}\frac{\eta\phi}{2} & (\boldsymbol{\mu}_X(X) \geq 1/\kappa), \\ \frac{\eta}{2}(1 - 2/\kappa) & (\boldsymbol{\mu}_X(X) < 1/\kappa). \end{cases} \tag{6.54}$$

Note that $\boldsymbol{\mu}_X(A) > 1/2$ and $\boldsymbol{\mu}_Y(B) > 1/2$ cannot hold simultaneously under the assumption $\alpha\boldsymbol{\mu}_X(A) + \beta\boldsymbol{\mu}_Y(B) \leq 1/2$. Integrating the results derived for all cases, we obtain

$$\frac{F_s(A,B)}{\boldsymbol{\mu}_+(A,B)} \geq \frac{\eta}{2}\min\left\{\frac{2}{\kappa}\phi,\ 1 - \frac{2}{\kappa}\right\}, \quad \forall \kappa > 2. \tag{6.55}$$

Note that this inequality holds for all A and B with $0 < \boldsymbol{\mu}_+(A \cup B) \leq 1/2$. In this way, we can get a series of lower bound of the bottleneck ratio, using different values of κ. And the supreme of these lower bounds remains a lower bound. It is easy to see that the supreme attains when $2\phi/\kappa = 1 - 2/\kappa$, leading to

$$\sup_{\kappa > 2} \min\left\{\frac{2}{\kappa}\phi,\ 1 - \frac{2}{\kappa}\right\} = \frac{\phi}{1+\phi}. \tag{6.56}$$

It follows that

$$\Phi_*(\mathbf{P}_+) \geq \frac{\eta}{2}\frac{\phi}{\phi+1}. \tag{6.57}$$

This completes the proof of the lower bound. Next, we show the upper bound, which is easier. Due to the definition of bottleneck ratio, for any given partition of $X \cup Y$, the flow ratio derived from that partition constitutes an upper bound of $\Phi_*(\mathbf{P}_+)$.

Here, we consider the partition with one part being X and the other being Y. Then

$$F_s = \alpha b = \beta f, \tag{6.58}$$

and thus the flow ratio is given by

$$\frac{F_s}{\min(\alpha, \beta)} = \max(f, b). \tag{6.59}$$

This gives an upper bound of $\Phi_*(\mathbf{P}_+)$. The proof of the theorem is completed. □

Laziness

Whereas increasing bottleneck ratio can enlarge the spectral gap, $1 - \lambda_2$, the mixing time also depends on $1+\lambda_n$, the distance between λ_n and -1. In general, a reasonable value of $1 + \lambda_n$ can be achieved by *laziness.*

Lemma 6.6. *Let* $\mathbf{P}$ *be a reversible transition matrix over* X*, such that* $\mathbf{P}(x, x) \geq \xi > 0$ *for each* $x \in X$ *then its smallest eigenvalue* λ_n *has* $\lambda_n \geq 2\xi - 1$.

This shows that by maintaining a probability $\xi > 0$ for the chain to stay (without transiting to other states), we can keep λ_n away from -1.

Proof. Let $\mathbf{P}' = (\mathbf{P} - \xi\mathbf{I})/(1 - \xi)$. Since $\mathbf{P}$ has $\mathbf{P}(x, x) > \xi$ for each $x \in X$, the entries of the matrix $\mathbf{P}'$ are all non-negative. In addition,

$$\mathbf{P}'\mathbf{1} = \frac{1}{1-\xi}(\mathbf{P} - \xi\mathbf{I})\mathbf{1} = \frac{1}{1-\xi}(\mathbf{1} - \xi\mathbf{1}) = \mathbf{1}. \tag{6.60}$$

This implies that $\mathbf{P}'$ is also a stochastic matrix. Since $\mathbf{P}$ is reversible, all its eigenvalues are real numbers. Without losing generality, we assume they are $\lambda_1 \geq \ldots \geq \lambda_n$. As $\mathbf{P}$ is a stochastic matrix, we have $\lambda_1 = 1$ and $\lambda_n \geq -1$. According to the spectral mapping theorem, the eigenvalues of $\mathbf{P}'$, denoted by $\lambda'_1, \ldots, \lambda'_n$, are given by $\lambda'_i = (\lambda_i - \xi)/(1 - \xi)$, for each $i = 1, \ldots, n$. As $\mathbf{P}'$ is a stochastic matrix, we have $\lambda'_n \geq -1$, and thus $\frac{\lambda_n - \xi}{1-\xi} \geq -1$. Therefore, $\lambda_n \geq 2\xi - 1$. The proof is completed. □

Given an arbitrary reversible chain with transition matrix $\mathbf{P}$, we can make it lazier by changing $\mathbf{P}$ to $(1-\xi)\mathbf{P}+\xi\mathbf{I}$. However, it is worth noting that increasing the *laziness coefficient* ξ would on the other hand shrink the spectral gap from $1-\lambda_2$ to $(1-\xi)(1-\lambda_2)$. Hence, it is advisable to select a ξ that balances laziness and spectral gap. Here, the optimal ξ that maximizes the absolute spectral gap is given by $\hat{\xi}=(\lambda_2+\lambda_n)/(\lambda_2+\lambda_n-2)$. When it is difficult to derive λ_2 and λ_n, one can use the estimates to set ξ.

6.3.4 Hierarchical Bridging Markov Chain

Based on the theory of bridging Markov chains, we develop practical algorithms to construct the bridges and sample from the joint chain.

Construction of Bridges

Come back to our original problem of sampling from a distribution $\boldsymbol{\mu}_X$ over X, for which we can get a Markov chain $\mathbf{P}_X$ based on local moves. To improve the mixing, we introduce "bridges" to facilitate non-local transition. Specifically, we first choose a collection of state subsets of X: $S_1,\ldots,S_m$, and create a bridging state y_i for each S_i. In this way, we get a set of new states $Y=\{y_1,\ldots,y_m\}$. Suppose each target state in X has been covered by some such subset Next, for each $x\in X$, we set a transition probability $\mathbf{Q}_B(x,y_i)=1/m(x)$ for each y_i associated with with it, *i.e.* $x\in S_i$, where $m(x)$ is the number of such bridges, and set $\mathbf{Q}_B(x,y_i)=0$ when $x\notin S_i$. According to Lemma 6.3, we can construct $\mathbf{Q}_F$, the transition probabilities from Y to X, as follows

$$\mathbf{Q}_F(y_i,x)=\boldsymbol{\mu}_X(x)/\sum_{x'\in S_i}\boldsymbol{\mu}_X(x'). \tag{6.61}$$

It is not difficult to verify that the matrices $\mathbf{Q}_B$ and $\mathbf{Q}_F$ as above satisfy the cross-space detailed balance in Eq.(6.21), with $\boldsymbol{\mu}_Y$ given by

$$\boldsymbol{\mu}_Y(y_i)\propto\sum_{x\in S_i}\boldsymbol{\mu}_X(x'). \tag{6.62}$$

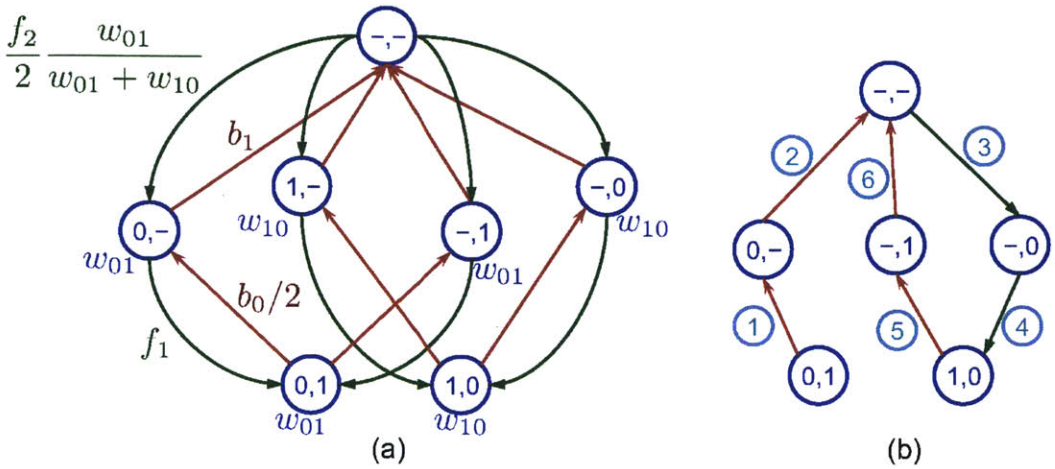


Figure 6-2: (a) shows the hierarchically bridging Markov chain on a simple problem: $x_1, x_2 \in \{0, 1\}$ with constraint $x_1 \neq x_2$. We use red color for the backward transitions from children to parents, and green for the transitions from parents to children. (b) illustrates a typical transition path. We use numbered circles to indicate the transition order. In this process, the bridges $(0, -)$ and $(-, -)$ are constructed upon the backward transition from a child state. When $(-, -)$ is instantiated, the right branch has not been visited, and the forward probability value for that branch is set with an optimistic estimate, encouraging the chain to visit that branch. Upon seeing $(1, 0)$, the forward probabilities of its parents will be updated accordingly.

The values of f and b are set empirically. The guideline is to keep a balance between the local updates along the original chain and the transition via bridges.

This construction is very flexible. Given a specific problem, one can choose the subsets in any way that they see as best. For example, for a problem where we have a clear perspective of the space structure, we can establish bridges that connect between the samples in different clusters to speed up the transition between them.

For problems with huge space, one layer of bridging can be very expensive. For such problems, we devise a novel sampling scheme called *hierarchical bridging*, which provides a systematic way to derive an ergodic chain.

Hierarchical Bridging

For many problems, the underlying clustering structure of the sample space is largely unknown, and thus it is difficult to devise the bridges in advance. In the following, we describe a generic approach, which extends the construction above to a hierarchical framework that recursively builds bridges at multiple levels.

Initially, we have the target state space X, where each sample is a discrete vector, in form of $(x_1, \ldots, x_K)$. The target states constitute the 0-th level of the hierarchy. For the first level of bridging, we introduce a set of bridges, denoted by Y_1. Each bridge in Y_1 corresponds to a partial assignment, *i.e.* a vector with one of the value removed. Take a state space $\{0,1\}^3$ for example. Consider $(0,0,0) \in X$. By removing the middle value, we get a partial vector $(0,-,0)$, where $-$ indicates a slot at which the value is removed. All vectors in form of $(0,x_2,0)$, which include $(0,0,0)$ and $(0,1,0)$ here, are called the *children* of $(0,-,0)$, and $(0,-,0)$, in turn, is called the parent of them.

Given $b_0, f_1 < 1$, the transition between X and Y is described as follows. Starting from a target state $x \in X$, with probability $1 - b_0$, the chain stays in X, and with probability b_0, it transits to the parent of x in Y_1, by randomly removing a value. Note that a vector of length K has K different parents, and thus the transition probability from x to any particular parent is b_0/K. Starting from a bridge $y \in Y_1$, with probability $1 - f_1$, it stays at y, and with probability f_1, it transits back to X. In particular, the transition probability from $y = (x_1, \ldots, -, \ldots, x_K)$ to $x = (x_1, \ldots, x_i, \ldots, x_K)$ is proportional to $\boldsymbol{\mu}(x)$. To calculate this probability, one only have to evaluate of $\boldsymbol{\mu}(x)$ up to a scale. This is a useful property, as the normalization constant of a distribution is often difficult to evaluate in practical problems.

The construction of the hierarchy can be completed by recursively adding levels up to the root (the K-th level). Each bridge at the k-th level (denoted by Y_k) is a partially assigned vector with k entries removed. Starting from $y \in Y_k$, the probability of transiting to the upper level Y_{k+1} is b_k. Specifically, each $y \in Y_k$ has $K - k$ assigned values, and thus it has a probability $b_k/(K-k)$ to transit to any of its parent by randomly removing one of the assigned values. The chain also has a total probability f_k to transit to the lower level Y_{k-1}. To accomplish such a transition, we randomly pick one of the k unassigned slots (say the j-th entry), and draws a value for x_j, resulting a child state y'. The forward transition probability from y to y' is proportional to $\boldsymbol{\mu}_{k-1}(y')$. For any bridge state $y \in Y_k$, the value $\boldsymbol{\mu}_k(y)$ is defined via

the recursive formula below

$$\boldsymbol{\mu}_k(y) \propto \sum_{y' \in Ch(y)} \boldsymbol{\mu}_{k-1}(y'). \tag{6.63}$$

When $k = 0$, $\boldsymbol{\mu}_0(x) \triangleq \boldsymbol{\mu}(x)$ for $x \in X$. Here, $Ch(y)$ is the set of y's children in Y_{k-1}. Through this construction, we obtain a joint chain over $X \cup Y_1 \cup \cdots \cup Y_K$, which we call the *hierarchically bridging Markov chain*, as illustrated in Figure 6-2(a). We derive the theorem below that characterizes this chain:

Theorem 6.5. *The hierarchically bridging Markov chain with $b_k < 1$ for $k = 0, \ldots, K-1$, and $f_k < 1$ for $k = 1, \ldots, K$ is ergodic. If we write the equilibrium distribution in form of $(\alpha\boldsymbol{\mu}_0, \beta_1\boldsymbol{\mu}_1, \ldots, \beta_K\boldsymbol{\mu}_K)$, then (S1) $\boldsymbol{\mu}_0$ equals the target distribution $\boldsymbol{\mu}$; (S2) for each $k \geq 1$, and $y \in Y_k$, $\boldsymbol{\mu}_k(y)$ is proportional to the total probability of its descendant target states (the target states derived by filling all its placeholders); (S3) α, the probability of being at the target level, is given by $\alpha^{-1} = 1 + \sum_{k=1}^{K}(b_0 \cdots b_{k-1})/(f_1 \cdots f_k)$.*

Here, we briefly explain the statements. (S1), together with the proved ergodicity, establishes the correctness of the construction, *i.e.* drawing samples from the joint chain and retaining only those from X amounts to directly sampling from $\boldsymbol{\mu}$. (S2) characterizes the distribution within other levels. (S3) gives the probability that a state drawn from the joint chain is a target state. From this statement, we derive

Corollary 6.2. *If $b_k/f_{k+1} \leq \kappa < 1$ for each $k = 1, \ldots, K$, then $\alpha > 1 - \kappa$.*

This lower bound of α is independent from the number of levels K. Consequently, despite the problem scale, one can maintain a considerable chance of drawing a target state from the joint chain by keeping the backward/forward ratio below 1.

This corollary can be easily shown as follows.

Proof. Based on Theorem 6.5, we have

$$\frac{1}{\alpha} = 1 + \sum_{k=1}^{K} \frac{b_0 \cdots b_{k-1}}{f_1 \cdots f_k} \leq 1 + \sum_{k=1}^{\infty} \kappa^k = \frac{1}{1-\kappa}. \tag{6.64}$$

Hence, $\alpha \geq 1 - \kappa$. The proof is done. □

Proof of Theorem 6.5

We show this theorem by progressively proving a series of claims as follows.

Claim 1. *The augmented Markov chain is ergodic.*

Proof. With $b_k > 0$ for $k = 0, \ldots, K-1$, the root is accessible from each state (including both complete and partial assignments). With $f_k > 0$ for $k = 1, \ldots, K$, each state is accessible from root. These imply that any two states are accessible from each other via the root. Therefore, the chain is irreducible. In addition, $f_K < 1$ makes the chain aperiodic. As this is a finite Markov chain, we can conclude that it is ergodic.

Since the chain is ergodic, it has a unique stationary distribution, *i.e.* its equilibrium distribution. Therefore, it suffices to show that $(\alpha\boldsymbol{\mu}_0, \beta_1\boldsymbol{\mu}_1, \ldots, \beta_K\boldsymbol{\mu}_K)$ that satisfies the three statements in the theorem is a stationary distribution. □

Claim 2. *Given vectors $\boldsymbol{\mu}_0, \ldots, \boldsymbol{\mu}_K$ respectively over the set of states at level $0, \ldots, K$, such that $\boldsymbol{\mu}_0 = \boldsymbol{\mu}$ is a distribution over X, and for each $k = 1, \ldots, K$, $\boldsymbol{\mu}_k$ is defined recursively by*

$$\boldsymbol{\mu}_k(y) = \frac{1}{K-(k-1)} \sum_{x \in Ch(y)} \boldsymbol{\mu}_{k-1}(x), \quad \text{for } y \in Y_k. \tag{6.65}$$

Then, $\boldsymbol{\mu}_k$ for each $k = 1, \ldots, K$ represents a distribution over Y_k, and

$$\boldsymbol{\mu}_k(y) = \binom{K}{k}^{-1} \sum_{x \in X : x \succ y} \boldsymbol{\mu}(x), \quad \forall y \in Y_k. \tag{6.66}$$

Here, $x \succ y$ means that x is a descendant of y.

Proof. Obviously, when $k = 1$, according to the definition above, we have

$$\boldsymbol{\mu}_1(y) = \frac{1}{K} \sum_{x \in Ch(y)} \boldsymbol{\mu}_X(x) = \frac{1}{K} \sum_{x \in Ch(y)} \boldsymbol{\mu}(x). \tag{6.67}$$

This satisfies Eq.(6.66), as $\binom{K}{1} = K$, and it is clear that $\boldsymbol{\mu}_1$ is non-negative. In addition, we have

$$\begin{aligned}\sum_{y \in Y_1} \boldsymbol{\mu}_1(y) &= \sum_{y \in Y_1} \frac{1}{K} \sum_{x \in Ch(y)} \boldsymbol{\mu}(x) \\ &= \frac{1}{K} \sum_{y \in Y_1} \sum_{x \in Ch(y)} \boldsymbol{\mu}(x) \\ &= \frac{1}{K} \sum_{x \in X} \boldsymbol{\mu}(x) \sum_{y \in Pa(x)} 1 = 1. \end{aligned} \tag{6.68}$$

Here, $Pa(x)$ is the set of parent states of x. In the derivation above, we use the fact that x has K parents, *i.e.* $|Pa(x)| = K$. The identity above implies that $\boldsymbol{\mu}_1$ is a valid distribution over Y_1. Therefore, the claim holds when $k = 1$.

Suppose that the claim holds for $k = 1, \ldots, m$ with $m < K$, we are to show that it also holds for $k = m + 1$, so as to complete the induction. Note that $\boldsymbol{\mu}_{m+1}$ is defined as

$$\boldsymbol{\mu}_{m+1}(y) = \frac{1}{K - m} \sum_{x \in Ch(y)} \boldsymbol{\mu}_m(x), \quad \text{for } y \in Y_{m+1}.$$

Again, $\boldsymbol{\mu}_{m+1}$ is obviously non-negative, and

$$\begin{aligned}\sum_{y \in Y_{m+1}} \boldsymbol{\mu}_{m+1}(y) &= \frac{1}{K - m} \sum_{y \in Y_{m+1}} \sum_{z \in Ch(y)} \boldsymbol{\mu}_m(z) \\ &= \frac{1}{K - m} \sum_{z \in Y_m} \boldsymbol{\mu}_m(z) \sum_{y \in Pa(x)} 1 = 1. \end{aligned} \tag{6.69}$$

Similar to the derivation for $k = 1$, here we apply the fact that $|Pa(x)| = K - m$ for each $x \in Y_m$. This shows that $\boldsymbol{\mu}_{m+1}$ is a valid distribution over Y_m. Moreover, we

have for each $y \in Y_{m+1}$,

$$\begin{aligned}
\boldsymbol{\mu}_{m+1}(y) &= \frac{1}{K-m} \sum_{z \in Ch(y)} \binom{K}{m}^{-1} \sum_{x \in X : x \succ z} \boldsymbol{\mu}(x) \\
&= \frac{1}{K-m} \frac{(K-m)!m!}{K!} \sum_{z \in Ch(y)} \sum_{x \in X : x \succ z} \boldsymbol{\mu}(x) \\
&= \frac{(K-m-1)!m!}{K!} (m+1) \sum_{x \in X : x \succ y} \boldsymbol{\mu}(x) \\
&= \binom{K}{m+1}^{-1} \sum_{x \in X : x \succ y} \boldsymbol{\mu}(x). \qquad (6.70)
\end{aligned}$$

By induction, we can conclude that the claim holds for each $k = 1, \ldots, K$. □

Claim 3. *When the construction is done up to the k-th level, the distribution* $\boldsymbol{\mu}_k^+ \triangleq (c_{k,0}\boldsymbol{\mu}_0, \ldots, c_{k,k}\boldsymbol{\mu}_k)$ *is a stationary of the augmented Markov chain. Here,* $\boldsymbol{\mu}_0, \ldots, \boldsymbol{\mu}_k$ *are given by Claim 2, and* $c_{k,0}, \ldots, c_{k,k}$ *is defined such that for each* $k' = 0, \ldots, k$

$$c_{k,0} = \frac{1}{Z_k}, \quad c_{k,l} = \frac{1}{Z_k} \frac{b_0 \cdots b_{l-1}}{f_1 \cdots f_l}, \qquad (6.71)$$

with

$$Z_k = 1 + \sum_{l=1}^{k} \frac{b_0 \cdots b_{l-1}}{f_1 \cdots f_l}. \qquad (6.72)$$

Proof. We are going to show this claim by induction. Note that $\boldsymbol{\mu}_0 = \boldsymbol{\mu}$ over X is a stationary distribution of $\mathbf{P}$. And $\boldsymbol{\mu}_0^+ = c_{0,0}\mathbf{p}_0$, thus $c_{0,0} = 1$. It immediately follows that the claim is true for $k = 0$. Suppose this claim holds for $k = 0, \ldots, m$ with $m < K$, we are to show that it also holds for $k = m + 1$. Note from Eq.(6.71) that

$$\frac{c_{m+1,k}}{c_{m,k}} = \frac{Z_m}{Z_{m+1}}, \quad \forall k = 0, \ldots, m, \qquad (6.73)$$

Hence, showing the claim holds for $k = m+1$ is equivalent to showing that $(\alpha\boldsymbol{\mu}_m^+, \beta\boldsymbol{\mu}_{m+1})$

is a stationary distribution of the augmented chain (up to $(m+1)$-th level), with

$$\alpha = \frac{Z_m}{Z_{m+1}}, \tag{6.74}$$

and

$$\beta = \frac{Z_{m+1} - Z_m}{Z_{m+1}} = \frac{1}{Z_{m+1}} \frac{b_0 \cdots b_m}{f_1 \cdots f_{m+1}}. \tag{6.75}$$

According to Lemma 6.3, it suffices to check that this distribution satisfies the cross-space detailed balance given in Eq.(6.21), which is not difficult to verify based on the construction described in section 3.2. □

In this proof, Claim 1 proves the ergodicity of the joint chain. Claim 2 constructs a set of vectors $\boldsymbol{\mu}_0, \ldots, \boldsymbol{\mu}_K$, and states that they are valid distributions over $X, Y_0, \ldots, Y_K$, and satisfy the properties given in (S2). Claim 3 (induction up to $k = K$) shows that the distributions constructed in Claim 2 is exactly a stationary distribution of the joint chain. Since the chain is ergodic, this is the equilibrium distribution. As a by product, Claim 3 also shows the the statement (S3) of the theorem. For (S1), it is automatically established by the construction described in Claim 2. Therefore, we can conclude that the proof of the theorem has been completed.

6.3.5 Dynamic Construction

Whereas the total number of bridges can be huge generally for a moderate problem, which however need not be explicitly instantiated prior to sampling. Instead, we can *build the chain progressively along with the sampling procedure.* As shown in Figure 6-2(b), except for the initial state that we start with, each state is instantiated only upon the first transition to it. In addition, we maintain references from each state to all its parents and children, to facilitate the transition from one state to another.

When a bridge state is constructed, one needs to determine the forward transition probabilities from this state to its immediate children. Exact evaluation of these probabilities requires complete knowledge of the distribution of all its descendants, which is generally unavailable upon the construction. A natural idea is to obtain such

information by recursively visiting all the descendants. However, the complexity of this method can grow exponentially as we travel up along the hierarchy, making it infeasible in practice.

To address this issue, we adopt a dynamic programming strategy. Consider a bridge y at the k-th level with a set of children $Ch(y)$. Recall that for each child state $y' \in Ch(y)$, the forward transition probability from y to y' is proportional to $\boldsymbol{\mu}_{k-1}(y')$. If y' has been visited, then $\boldsymbol{\mu}_{k-1}(y')$ is immediately available. Otherwise, we initially use a quick estimate of $\boldsymbol{\mu}_{k-1}(y')$ and update it when y' and its descendants are visited.

In general, one can overestimate the forwarding probability of an unvisited branch, thereby encouraging exploration of unknown regions. The initial value need not be accurate, as it is updated as the branch below y' is visited. A possible way to this quick estimation is to assume all assignments in that branch are valid (*i.e.* satisfying all constraints). For both applications described in next section, we employ this way, which results in an estimate as the product of the marginal probabilities of the available values.

In this scheme, the transition probabilities can change dynamically, resulting in time-inhomogeneity. In practice, such changes to the chain happen primarily during burn-in, and thus have negligible effect on asymptotic behavior. It is also worth noting that while the total number of states in X can be tremendous even for a problem with moderate size, our algorithm generally only visit those states with non-negligible probabilities. Though just constituting a small fraction of the entire space, they still provide a close approximation of the target distribution.

6.4 Experiments

We assess the effectiveness of the proposed method on both synthetic and real data. Specifically, we first test it on a constrained binary labeling problem, where the proposed sampling algorithm is used to sample from the solution space subject to a set of synthetic constraints. As the ground-truth is available for this problem, we can

perform a systematic study through this set of experiments.

We also test this approach on layered modeling for images and videos. The aim here is to demonstrate both the effectiveness of the sampling method in sampling partial orders and the the use of partial orders combined with our sampling approach improves the solution.

6.4.1 Constrained Binary Labeling

Given a graph with n nodes and m edges, we are to set a binary label $x_i \in \{0, 1\}$ to each node. Here, each edge is associated with a constraint on the labels of its two ends (*e.g.* $x_i \neq x_j$). We use an n-dimensional vector $x \in \{0, 1\}^n$ to represent a label configuration, and use Ω to denote the set of all configurations that satisfy the constraints. In addition, each node has a preference function $w_i : \{0, 1\} \to \mathbb{R}^+$. Then, we get a distribution over Ω, given by $p(x) \propto \prod_{i=1}^{n} w_i(x_i)$. While the probabilities are in a product form, the labels are not independent as they are related to each other via the constraints. This formulation actually stems from real world problems, such as circuit design, scheduling, and object placement.

We first consider a 4-connected graph with 5×5 nodes. Though the graph might seem small, it is sufficient to generate a large enough state space (up to 2^{25}), where the differences of algorithm behaviors can be clearly seen. Importantly, with this scale, it is feasible to evaluate the entire distribution through enumeration, enabling direct comparison between the sample distribution and the true one. To obtain a constrained problem, we randomly draw a constraint for each edge from a set of constraints ($x_i = x_j$, $x_i \neq x_j$, $x_i = 1$ or $x_j = 1$, etc). In this way, we generate a set of 20 constrained labeling problems as a testbed.

On these problems, we compare three algorithms:

1. *Gibbs sampling with long jump (GS-Jump)*: a method adapted from the one proposed by [9]. At each iteration, we update all variables by Gibbs sampling, and then propose a jump to arbitrary configuration drawn from the product distribution, accepting it if the result is valid.

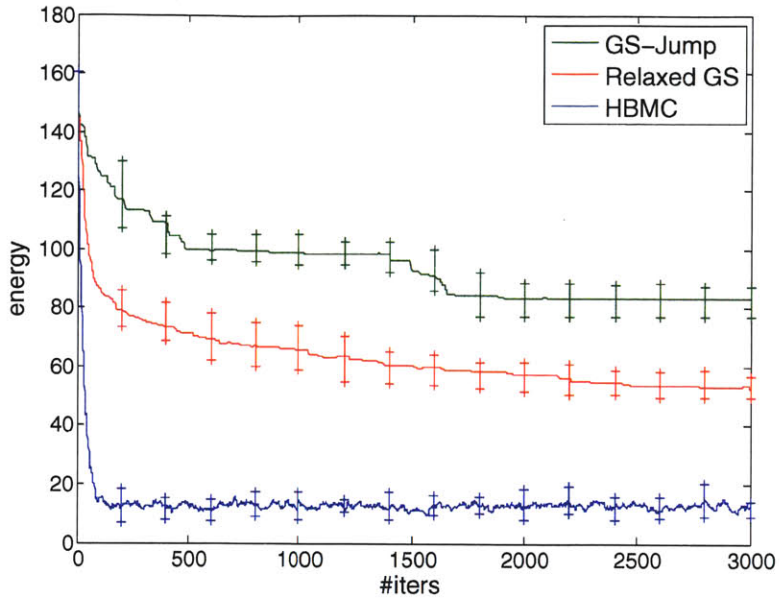


Figure 6-3: Each curve shows the mean energy values ($-\log p(x)$) as a function of elapsed iterations. Since Relaxed-GS and HBMC may yield states that are not in Ω, we use the energy of the last valid state as the energy value for an iteration. This also shows bars at 10% and 90% quantiles for 100 repeated runs.

2. *Relaxed Gibbs sampling (Relaxed-GS)*: similar to WalkSAT [111, 54], we modulate the probability with a factor $\exp(-c \cdot \#\{\text{violated constraints}\})$, and turn the constrained model into an unconstrained MRF, upon which Gibbs sampling is applied. Here, c is empirically set to balance approximation accuracy and sampling efficiency.

3. *Hierarchical Bridging Markov Chain (HBMC)*: this is our approach. Here, we set $b_0 = 0.5$, meaning that starting from a target state, the chain performs a Gibbs update with 50% chance, and transits to upper level with 50% chance. For all other levels, we set $b_k = 0.4$ and $f_k = 0.6$. Each iteration consists of 25 walks, just like the other methods in comparison.

Figure 6-3 compares the energy trajectories obtained from 100 independent runs on a constrained problem as described above. We can see that GS-Jump gets stuck locally before a long jump is accepted, which rarely happens (once per over 1000 iterations on average). By allowing violation of constraints with cost, Relaxed-GS escapes from local traps, though rather slowly. HBMC significantly outperforms the other

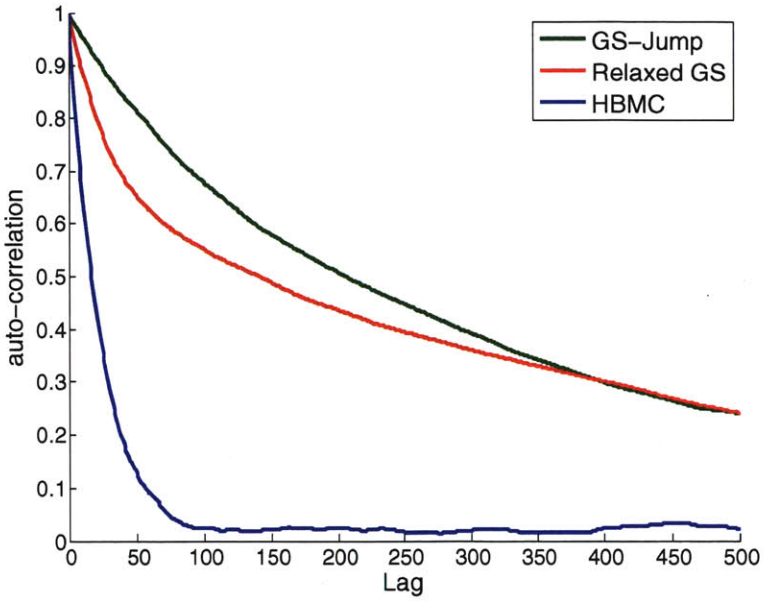


Figure 6-4: The energy auto-correlation function.

methods. Initially, encouraged by the optimistic weights set for unseen branches, the HBMC sampler quickly travels over the sample space, and at the same time builds bridges at different levels. In this process, the forward probabilities will be updated, with small values set to the branches leading to unlikely regions. Consequently, the chain rapidly gets to the states with high probabilities and rarely travels away.

Using the energy trajectories, we compute the *autocorrelation function*, averaged over all runs on all problem sets (in total 2000 runs for each algorithm). The results are shown in Figure 6-4. For HBMC, the correlation decreases to 0.1 after 50 iterations, and samples obtained with an interval of 80 can be considered as independent. Significant correlation remains for the other two methods even after 500 iterations, indicating that the underlying chains mix slowly.

We also investigate how many samples are needed to approximate the underlying distribution. For this study, we choose a constrained problem of which the number of distinct samples is about $10,000$, and collect $50,000$ samples for each algorithm, each per 200 iterations. We compute the correlation between the empirical distribution $\tilde{\mathbf{p}}$ and the true distribution $\mathbf{p}$, *i.e.* $\mathbf{p}^T\tilde{\mathbf{p}}/\sqrt{\|\mathbf{p}\|\|\tilde{\mathbf{p}}\|}$. The results are shown in Figure 6-5. The sample distribution obtained via HBMC is significantly closer to the true

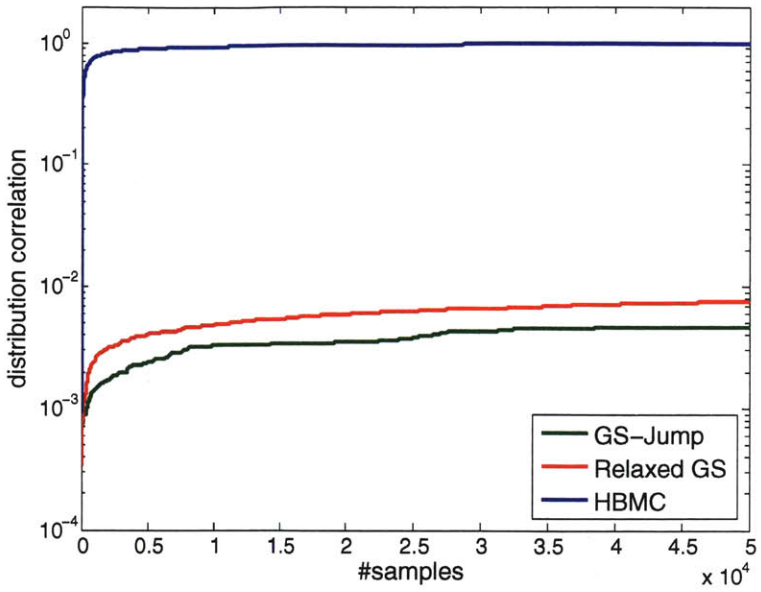


Figure 6-5: The correlations between the empirical distributions of the collected samples and the true distribution. Note that the y-axis is at log-scale.

distribution as compared to the other methods, obtaining a correlation of 0.9 after only 5,000 samplers. The other two methods exhibit drastically slower behavior. After 10 times greater samples, they remain stuck in a low-probability region with the correlation below 0.01.

6.4.2 Inferring Layer Orders from Synthetic Images

Next, we test the proposed method on the inference of the relative depth-order of visual layers. Specifically, we first performed a series of controlled experiments on synthetic images. To generate each image, we superimpose a set of templates in a random order and add white noise to it. We compared four methods for partial order inference, with the templates and their domains given to each method being tested.

1. *MRF:* directly estimates the indicator map of layers, with an MRF prior to enforce smoothness.

2. *BLK:* groups all pixels into blocks, with each covered by the same set of layers. It infers one top layer for each block.

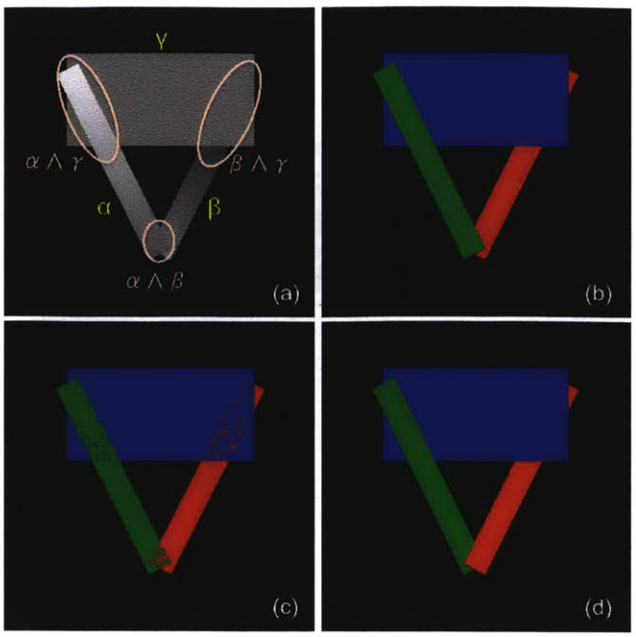


Figure 6-6: An illustrative example: (a) synthetic image with markups (this image + noise of $\sigma = 0.2$ is the input), (b) ground-truth (HBMC obtains this in most cases), (c) a result via MRF, (d) a result via BLK.

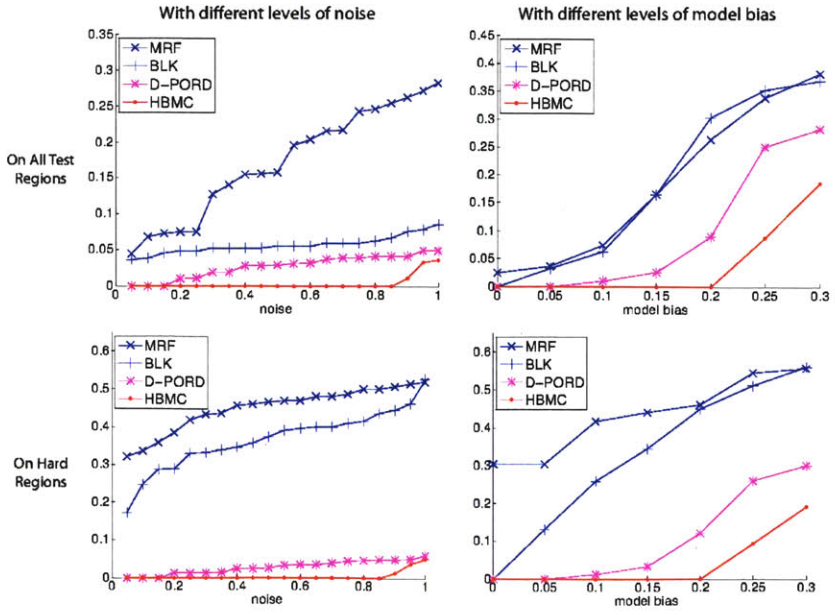


Figure 6-7: The average ratios of error labels on both test regions and hard regions over all 200 synthetic images, obtained using four methods under different levels of noise and model bias.

3. *D-PORD*: progressively determines the most probable top layer for each block, starting from the ones with high confidence. This method is based on our graph representation, and thus respects the partial order constraints. It is the best-performing heuristic method in this context.

4. *HBMC*: our sampling method based on a hierarchically bridging Markov chain.

The forward and backward probabilities are set to $f = 0.7$ and $b = 0.3$. We initialize the chain with the D-PORD result, take the first 500 steps as burn-in, and then collect $20,000$ samples at the interval of 50 steps. To verify our method, we computed the exact posterior distributions on a set of cases with $K \leq 5$. The resulting sample distributions are very close to the exact ones with L1-distance at about 10^{-3}.

Figure 6-6 shows the results on a synthetic image specially made for illustration. The input image, corrupted by white noise, contains three rectangles with order $\beta < \gamma < \alpha$. Most samples ($> 99\%$) obtained via HBMC are identical to the ground-truth. Without utilizing the ordering constraint, the MRF relies on local pixel values and smoothness, which are often ambiguous, leading to noisy labeling. Here, we have tuned the weight of smoothness terms to yield best overall performance. BLK generally performs better than MRF by making decisions based on entire blocks. However, the ambiguous block ($\alpha \cap \beta$) is labeled incorrectly 20% of the time, something which could be easily resolved by incorporating partial order and using the knowledge $\alpha > \gamma$ and $\gamma > \beta$ from other blocks.

We also performed systematic comparison between these methods. Specifically, we use templates of different types such as people, animals, and vehicles to synthesize a collection of 200 images. Each image is generated by superimposing some randomly positioned objects in a random Z-order. We then apply each method to infer the layer map. The testing was done with different levels of white noise. To reflect the common problem that the appearance templates are in themselves biased, we further added a random bias to the templates provided to the inference algorithm. The performance was measured by the ratio of error labeling over *test regions*, the ones covered by

more than one layers.

The left column in Figure 6-7 shows the performance under different levels of noise, with σ_{noise} ranging from 0.05 to 1 (the dynamic range of pixel values is $[0, 1]$), while the right column shows the performance under different levels of appearance model bias, with σ_{bias} from 0.05 to 0.3. The first row shows the average error rate over all test regions in all images. To make the distinction clearer, we identified those regions of which the average differences between two closest covering templates are below 0.1 as *hard regions*, and show the average error rates over them in the second row.

The results show: (1) The methods utilizing block-constraints (i.e. the pixels in the same block have the same top layer) exhibit much better robustness against noise. (2) The methods based on partial orders (D-PORD and HBMC) consistently and significantly outperform others, subject to both noise and model bias, as the consistency constraints effectively coordinate the labeling across different blocks. (3) HBMC, which derive the results by summarizing from $20,000$ samples, is much more robust than D-PORD. It yields perfect performance (i.e. 0% errors), under moderate noise and model bias.

6.4.3 Inferring Layer Orders from Real Videos

To assess its practical utility, we applied our method to solve a real world problem, namely inferring the partial Z-order of cars in a 10-minute long video of a busy avenue. The focus here is on sampling partial orders, rather than developing a full-fledged video model, and therefore we employ simple approaches for motion and appearance modeling.

Specifically, we treat each car as an object layer, with a rectangular domain, and use Kalman filtering to update the positions of the cars and their templates. The Z-order is re-inferred each time based on the updates, using the previous Z-order as a prior.

Part of the results are shown in Figure 6-8, which shows that our method performs very well in inferring the partial Z-orders, despite the simplicity of the motion and

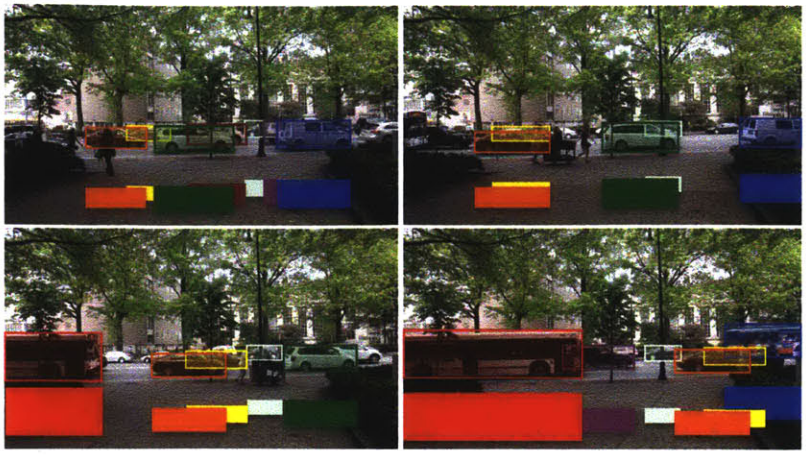


Figure 6-8: The inferred partial orders of vehicles in 4 frames of a video (interval = 3 sec). Vehicles are marked with transparent rectangles in different colors. Below them are opaque blocks that illustrate their Z-orders.

appearance models.

Implementation and Efficiency

In our C++ implementation, two techniques are used to accelerate the sampling process: (1) caching visited states together with their transition tables in a hash map, and (2) dynamically rearranging the entries of the transition table in descending order of probabilities (We will publish our code).

The sampler runs very efficiently. For a typical layered video model with 8 to 10 layers, it makes about 10 million Markov transition steps per second. Hence, using our scheme ($20,000$ samples with 50 steps each), it takes 0.1 second to perform inference.

6.5 Summary

We developed a new representation of partial order based on minimal sufficiency, and a principled approach for sampling from a constrained combinatorial space. The method provides a general way to address the difficulty arising from disconnected state spaces. The experiments demonstrated that our approach is effective and efficient in sampling from the posterior distribution of partial orders, and that explicit utilization of partial order can remarkably improve the robustness of the layered video model.

Chapter 7

Conclusions

In this work, we studied the modeling of dynamic visual scenes using a generative Bayesian approach, particularly focusing on three key aspects: *appearance*, *motion*, and *the depth order between layers*. we developed a series of machine learning techniques to address the challenges arising as a consequence of the model formulation and associated probabilistic inference. Included among these developments were a new construction of dependent Dirichlet processes and a new method to sample from constrained combinatorial spaces.

While does not yet achieve the ultimate goal of providing a complete and unified interpretation of visual scenes, this work does demonstrate the great potential of probabilistic generative models in vision applications. Currently, discriminative methods dedicated to specific tasks dominate the field of computer vision, probably due to the fact that they showed good performance on numerous real world applications. However, the task-oriented nature of these methods makes it difficult, if not impossible, to bring them together to interpret the visual world that we see every day in a coherent fashion.

The exploration of generative approaches in this thesis is motivated by our deep belief in the value of generative models for computer vision. The main advantage of generative models is not that they perform better than discriminative methods on specific tasks, but that they provide a flexible and elegant framework to incorporate prior knowledge of different types and to integrate models of different structures

without comprising the mathematical rigor.

Using generative modeling, one may easily leverage the relations between different aspects of a problem, and thus derive better solutions, addressing issues that would otherwise be difficult to resolve. For example, as we have shown in Chapter 3, the desire to express rich structures of local patterns and the need to ensure global coherence can be coupled elegantly through a generative MRF model. Moreover, in Chapter 5, we showed that the construction of mixtures whose number of components may change over time can be accomplished using a nonparametric generative formulation. However, to our best knowledge, discriminative methods that solve such problems in a satisfactory way have not existed yet.

7.1 Summary of Contributions

A major contribution of this work is the development of a Bayesian generative framework that integrates the modeling of appearance, motion, and the depth order of layers. In addition, we made a series of significant contributions to various topics, as summarized below.

A new image model

To describe images and video frames, a new image model is developed in Chapter 3, which integrates a probabilistic manifold model that can express a rich set of local structures with a Markov random field that enforces coherence across local patches. An important aspect of this formulation that distinguishes it from other high-order MRF models, such as the Field of Experts [83], is that the likelihood that a patch is generated from the patch manifold is used as the potential functions. This new design of potential functions offers much greater capability of expressing local structures, as compared to those based on linear filter responses. In addition, through the overlapping of local patches, the joint MRF encourages the global coherence of images.

A new model of persistent motion patterns

Dynamics plays an important role in visual scene understanding. Whereas extensive study has been devoted to accurate estimation of local velocities (*e.g.* object tracking and optical flow), capturing persistent motion patterns over a region and during a period may be more important in terms of understanding the activity underlying a given scene. To effectively model persistent motion patterns, a new notion called the *geometry flow* is introduced in Chapter 4, which unifies two different types of motion characterizations, particularly an ensemble of trajectories and a continuous transformation process. A Lie algebraic representation is then derived, which maps each flow to an element in a vector space, thus greatly simplifying both probabilistic formulation and flow estimation from observed scenes.

A new non-parametric construction of dynamic mixture models

Mixture models are used in both the image model and motion model to capture complex distributions. For example, the patch manifold of the image model uses a mixture distribution to describe local patterns, and the motion model uses a mixture of flows to describe the motion patterns in a complex scene. In the past decade, non-parametric mixture models based on Dirichlet processes were developed and became popular. Such models allow for an indefinite (countably infinite) number of components and as such, provide a flexible mechanism to construct mixture models when the number of components is difficult to specify a priori.

To model dynamic scenes, we desire a mixture model which may evolve over time. The key challenge here is to incorporate temporal dependency between mixture models at different time steps. To address this problem, we developed a new construction of dependent Dirichlet processes in Chapter 5. By exploiting the inherent theoretical connections between the Poisson and Dirichlet processes, this construction allows dynamically creating and removing components and varying the component parameters, while guarantees that the marginal distribution of components at each time step is itself a Dirichlet process.

A new method to infer partial orders

Our framework used layers to model visual scenes with multiple objects. A challenge arising here is to determine the occlusion relation between layers, which, in turn, is determined by their relative depth order. Layered models developed in previous work either uses ad-hoc methods to estimate this order, or circumvents the problem by directly inferring the association between pixels and layers. Neither approach guarantees consistent occlusion reasoning.

We explored a generative approach to this problem in chapter 6, that is, to formulate a prior over the partial order between layers, and estimate it through inference. A difficulty here is that a partial order has to satisfy a set of combinatorial constraints (*e.g.* anti-symmetry and transitivity), and it is difficult to perform inference while preserving these properties. In tackling this difficulty, we developed a new method for MCMC sampling, which dynamically introduces virtual states that bridge different parts of the space that would otherwise be disconnected due to the combinatorial constraints. Though motivated by a specific vision problem, this is a generic methodology that can be applied to solve other inference problems over combinatorial spaces.

7.2 Future Directions

As mentioned, the work presented in this thesis shows the great potential of generative modeling in vision applications. However, many problems are yet to be solved. Next, we briefly discuss several future directions to extend this work that we feel are important and deserve further efforts.

Modeling dynamic shapes

One aspect that has not been addressed in this work is the modeling of shapes, which, in our framework, determines the domain of each layer. Whereas there has been extensive work on shape modeling, including active contours, level set methods, and deep models, generative modeling of shapes, especially in a dynamic context, remains a nontrivial challenge. In addition, it is also interesting to study how to

model the interactions between shapes and motion, and how to perform inference on shapes when part of an object is occluded.

Semantic interpretation

The appearance and motion models establish an intermediate representation of visual scenes, which, however, does not directly provide a semantic interpretation of the scene. An additional layer is needed to connect between this intermediate representation and the semantics (*e.g.* the categories of objects and scenes, and their relations). A possible approach to this problem is to consider a scene as composed of parts coming from different *topics*, each associated with its own appearance and motion model. To compose a scene, the spatial arrangement of these topics should follow some regular patterns, which, again, can be captured using a generative model.

Adaptation for discriminative tasks

Though a generative model can be formulated and trained in a task-independent way, it is often desirable to adapt the model to the target context when it is applied to a specific task, especially the discriminative tasks (*e.g.* object recognition and scene classification). We believe using specific knowledge about a particular context to adapt a generic model not only increases the accuracy and efficiency, but also helps to address ambiguities that would need specific contextual information to resolve.

Structured non-parametric models

Bayesian nonparametrics offers an elegant and powerful means to construct mixture models without the need to specify the number of components in advance. However, it is nontrivial to incorporate statistical dependencies between nonparametric models. Previous efforts, including Hierarchical DP, Hierarchical Beta Process, and one of the contributions of this thesis on Poisson-based Dependent DP, provide useful ways to construct dependent nonparametric models. Yet, these are limited in several respects, most notably the following: they can only be applied to the case where the dependency graph has a tree structure, and there are no interactions between components.

The need to capture more complicated dependencies often arises in vision problems. Solving these problems requires the development of new nonparametric models. Generally, there are several approaches to accomplish this: (1) introducing auxiliary stochastic processes to help establish dependencies, (2) generalizing the point processes that underlie the Dirichlet processes, or (3) generalizing the stick breaking processes. We believe that these approaches are related to each other, and the analysis of their relations may lead to the development of a more generic family of nonparametric models.

Appendix A

Basics of Group Theory

A.1 Basic Concepts of Group

Group is a fundamental concept in modern algebra, which has played a significant role in various fields in mathematics, physics, and computer science.

Definition A.1 (Group). *A group $(G, \cdot)$ is defined as a set G with a binary operation $\cdot$, which satisfies the following axioms:*

1. *The group is closed under the product operation $\cdot$, i.e. $\forall x, y \in G,\ a \cdot b \in G$.*
2. *The product operation satisfies associativity: $\forall x, y, z \in G,\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$.*
3. *There exists an identity element e, such that $\forall x \in G,\ e \cdot x = x \cdot e = x$.*
4. *For each element $x \in G$, there exists an inverse element $x^{-1} \in G$, such that $x \cdot x^{-1} = x^{-1} \cdot x = e$.*

It can be easily shown that *the identity element of a group is unique*, and for each element in the group, *its inverse element is unique.*

In addition, the inverse operation has the following properties. For arbitrary

elements x, y in a group G,

$$(x \cdot y)^{-1} = y^{-1} \cdot x^{-1},$$
$$(x^{-1})^{-1} = x,$$
$$e^{-1} = e.$$

Definition A.2 (Abelian Group, Commutative Group). *A group G in which the product operation satisfies commutativity, i.e. $\forall x, y \in G, x \cdot y = y \cdot x$, is called an Abelian group, or a commutative group.*

Abelian groups have a lot of nice properties that we will discuss later.

Definition A.3 (Symmetric Group). *The symmetric group of a set X, denoted by* $\mathrm{Sym}(X)$, *it the group consisting of all bijective mappings from X to X with the product operation being the composition of mapping. Let $f, g \in \mathrm{Sym}(X)$, then $\forall x \in X$, $(f \cdot g)(x) = f(g(x))$.*

The identity element of a symmetric group is the identity map, that maps each element to itself.

When X is a finite set, its symmetric group comprises all the permutations, which is thus called a *permutation group*. When X is a continuous space, $\mathrm{Sym}(X)$ is conventionally called a *transformation group*.

Definition A.4 (Subgroup). *A subgroup of a group G is a subset H which is closed under the group operation, i.e. it satisfies the following properties*

1. *H is closed under product operation, i.e. $\forall x, y \in H$, $x \cdot y \in H$.*
2. *H contains the identity element, $e \in H$.*
3. *H is closed under inverse operation, i.e. $\forall x \in H$, $x^{-1} \in H$.*

Actually, the three properties above is equivalent to the following: $\forall, x, y \in H, x^{-1}y \in H$.

Obviously, for each group G, $\{e\}$ and G itself are subgroups of G, which are called *trivial subgroups*. Other subgroups are called *nontrivial subgroups*.

Given any subset S of a group G, *the subgroup generated by* S is the smallest subgroup that contains S, which comprises exactly all the products of elements in S and their inverses.

Definition A.5 (Product Group, Direct Product). *Let G and H be groups, their product group, also known as direct product, denoted by $G \times H$, consists of the Cartesian product of their underlying sets $\{(g, h) | g \in G \text{ and } h \in H\}$, with the product operation defined as*

$$(g_1, h_1) \cdot (g_2, h_2) = (g_1 \cdot g_2, h_1 \cdot h_2).$$

The element of the product group is (e_G, e_H), in which e_G and e_H are respectively the identity elements of G and H. And the inverse operation is thus defined by

$$(g, h)^{-1} = (g^{-1}, h^{-1}).$$

The definition above can be extended to the product of multiple groups straightforwardly.

A.2 Group Homomorphisms and Kernels

We can define maps between groups, among which homomorphisms are of particular interests.

Definition A.6 (Homomorphism). *Let G and H be groups. A map $\Phi : G \to H$ is called a homomorphism if it preserves product operations, i.e. $\forall x, y \in G$, $\Phi(x \cdot y) = \Phi(x) \cdot \Phi(y)$.*

It can be easily shown that the group homomorphism also has the following properties

1. It maps identity element to identity element, i.e. $\Phi(e_G) = e_H$.

2. It preserved inverse operation, i.e. $\forall x \in G, (\Phi(x))^{-1} = \Phi(x^{-1})$.

If Φ is a bijective map, then Φ is called an *isomorphism*. An isomorphism of a group onto itself is called an *automorphism*. Note that the set comprising all automorphisms of a group G together with the composition as product operation is also a group (a symmetric group), denoted by $\mathrm{Aut}(G)$.

Two groups G and H is said to be *isomorphic*, if there exists an isomorphism between them.

Definition A.7. *Let G and H be groups, and $\Phi : G \to H$ be a homomorphism, then the kernel of G, denoted by $\ker G$, is defined as $\{x \in G | \Phi(x) = e_H\}$, in which e_H is the identity element of H.*

It can be easily shown that $\ker \Phi$ is a subgroup of G. The kernel of an isomorphism is $\{e_G\}$.

A.3 Normal Subgroups and Quotient Groups

Definition A.8 (Coset). *Given a subgroup H of a group G, the left coset of $x \in G$ is $xH = \{xh | h \in H\}$, and the right coset of x is $Hx = \{hx | h \in H\}$.*

The set of all left cosets forms a partition of G, i.e. two left cosets are either equal or disjoint. The same applies to the set of all right cosets.

Given an arbitrary subgroup H, the left and right cosets may or may not be equal. If they are equal, H is called a *normal subgroup*.

Definition A.9 (Normal Subgroup). *A subgroup N of a group G is called a normal subgroup, denoted by $N \lhd G$, if it is invariant under conjugation, i.e. $\forall n \in N, x \in G$, $xnx^{-1} \in N$.*

For a subgroup N of a group G, the statement that *N is a normal subgroup* is equivalent to the following

For each $x \in G$, $xN = Nx$, in other words, left cosets and right cosets are equal. Hence, for a normal subgroup N, the left coset and right coset are both simply called the coset.

Not every subgroup of a group is normal. However, all subgroups of an Abelian group are normal. And, trivial subgroups are always normal.

Let N be a normal subgroup of G, we can define equivalence between any two elements in G as

$$x \sim y \Leftrightarrow xy^{-1} \in N.$$

With this equivalence, the equivalent class of $x \in G$, is

$$[x] = xN = Nx,$$

which is exactly the coset of x.

In addition, the equivalence has the following properties.

1. the equivalent class (coset) of the identity element is the normal set, i.e. $[e] = N$.
2. equivalence is preserved under product operation, i.e. if $x_1 \sim x_2$ and $y_1 \sim y_2$, then $x_1 \cdot y_1 \sim x_2 \cdot y_2$.
3. equivalence is preserved under inverse operation, i.e. if $x_1 \sim x_2$, then $x_1^{-1} \sim x_2^{-1}$.

From these properties, we can see that the cosets in themselves constitute a group, called the *quotient group.*

Definition A.10 (Quotient Group)**.** *Let N be a normal subgroup of a group G, the quotient group, denoted by G/N, is the set of all cosets with respect to N, with the product operation defined by*

$$[x] \cdot [y] = [x \cdot y].$$

In addition, it can be easily shown that the identity element in G/N is $[e] = N$, and the inverse operation can be given as

$$[x]^{-1} = [x^{-1}].$$

Normal subgroups and homomorphisms have close relations.

Theorem A.1. *Let G and H be groups, and $\Phi : G \to H$ be a homomorphism, then* $\ker \Phi$ *is a normal subgroup of G.*

Conversely, Let N be a normal subgroup of G, then the map: $\Phi : G \to G/N$ that maps each element to its coset, i.e. $\Phi(x) = [x] = xN$, is a homomorphism, whose kernel is N.

The above theorem establishes the correspondence between normal subgroups and homomorphisms.

One of the particular important normal subgroup of a group is its center.

Definition A.11 (Group Center). *The center of a group G is the set $Z(G)$ that consists of the elements that commute with all elements in G, i.e.*

$$Z(G) = \{z \in G | g \cdot z = z \cdot g, \ \forall g \in G\}$$

It can be easily seen that

$$\forall z \in Z(G), \forall g \in G, \ g \cdot z \cdot g^{-1} = z.$$

Hence, $Z(G)$ is a normal subgroup of G. And, it is obvious that $Z(G)$ is an Abelian group.

Let G be an Abelian group, then the center of G is G itself. At the other extreme, a group is said to be *centerless* if its center is the trivial group $\{e_G\}$.

Definition A.12 (Inner Automorphism Group). *The map $\Phi : G \to \mathrm{Aut}(G)$ which maps each element $g \in G$ to the corresponding conjugation ϕ_g defined by $\forall x \in G$, $\phi_g(x) = gxg^{-1}$. The range of Φ is called the inner automorphism group, denoted by* $\mathrm{Inn}(G)$.

Consider When $g \in Z(G)$, we can easily see that $\phi_g(x) \equiv x$. Actually, it can be shown that $Z(G)$ *is exactly the kernel of* Φ, i.e. $G/Z(G)$ is isomorphic to $\mathrm{Inn}(G)$.

A.4 Semidirect Product

Direct product discussed above is an elementary way to construct groups by integrating component groups. However, many important groups are formed with its generalization, called *semidirect product*. A semidirect product group also uses the Cartesian product as underlying set, but with a generalized multiplication operation.

Definition A.13 (Semidirect Product)**.** *Let G be a group, N be a normal subgroup of G ($N \lhd H$), and H be a subgroup of G. Then G is said to be a semidirect product of N and H, if*

$$G = NH \text{ and } N \cap H = \{e\}.$$

This condition is equivalent to any one of the following.

1. $G = HN$ *and* $N \cap H = \{e\}$,
2. *each element in G can be written uniquely in form of $n \cdot h$ with $n \in N$ and H.*
3. *each element in G can be written uniquely in form of $h \cdot n$ with $h \in H$ and $n \in N$.*

If this case, we say G splits over N.

Let G be a semidirect product of a normal subgroup N and a subgroup H, then we have

1. In each equivalence class in the quotient group G/N, there exists a unique element in H.
2. H is isomorphic to G/N, a natural isomorphism is to map each element h to $[h] = hN = Nh$.
3. The map $G \to H$ that takes each element $x \in G$ to the unique element in $H \cap [x]$ is a homomorphism. It can be seen that the map is an identity map on H and its kernel is N.

Definition A.14 (Normal Factor Semidirect Product w.r.t. Group Homomorphism). *Let N and H be two groups, and $\phi : H \to \mathrm{Aut}(N)$ be a group homomorphism. Then the left normal factor semidirect product of N and H with respect to ϕ, denoted by $N \rtimes_\phi H$, is a group with the Cartesian product $N \times H$ as the underlying set, and the multiplication defined as*

$$(n_1, h_1) \cdot (n_2, h_2) = (n_1 \cdot \phi_{h_1}(n_2),\ h_1 \cdot h_2)$$

Hence, the identity element is (e_N, e_H), and the inverse operation is given by

$$(n, h)^{-1} = (\phi_{h^{-1}}(n^{-1}), h^{-1}).$$

Likewise, we can define the right normal factor semidirect product of N and H with respect to ϕ, denoted by $N \ltimes_\phi H$ as a group with $H \ltimes N$ being the underlying set, and the multiplication defined as

$$(h_1, n_1) \cdot (h_2, n_2) = (h_1 \cdot h_2,\ n_1 \cdot \phi_{h_1}(n_2)).$$

It is obvious that if ϕ is a trivial homomorphism that sends each h to the identity map of N, i.e. $\phi_h \equiv \mathrm{Id}_N$, then the normal factor semidirect product degenerates to direct product.

In $G = N \rtimes_\phi H$, the pairs (n, e_H) form a subgroup of G that is isomorphic to N, while the pairs (e_N, h) form a subgroup of H that is isomorphic to H. Similar results can also be obtained for right normal factor semidirect product.

Theorem A.2. *Let G be a semidirect product of its normal subgroup N and another subgroup H, i.e. $G = NH$ with $N \cap H = \{e\}$. Then G is isomorphic to $N \rtimes_\phi H$ and $H \ltimes_\phi N$ for some group homomorphism ϕ.*

One example is the ϕ that maps each $h \in H$ to $\phi_h \in \mathrm{Aut}(N)$ defined by

$$\phi_h(n) = h \cdot n \cdot h^{-1}.$$

Appendix B

Basics of Differential Geometry

B.1 Basic Concepts of Manifolds

Manifold is a fundamental concept in modern geometry, which is established based on topology.

Definition B.1 (Topological Manifold). *A topological space M is said to be a topological manifold of dimension n, or a topological n-manifold, if it satisfies the following properties*

1. *M is a Hausdorff space, i.e. each pair of distinct points in M have disjoint neighborhood.*
2. *M is second countable, i.e. there exists a countable basis for the topology of M.*
3. *M is locally Euclidean, i.e. each point in M has a neighborhood that is homeomorphic to an open subset of $\mathbb{R}^n$.*

Definition B.2 (Chart). *Let M be a topological n-manifold, a coordinate chart on M is a pair (U, φ), in which U is an open subset of M, and $\varphi : U \to \mathbb{R}^n$ is a homeomorphism. Here, U is called the coordinate domain, while φ is called a (local) coordinate map. For each $p \in U$, $\varphi(p) = (x^1(p), x^2(p), \ldots, x^n(p))$ is called the local coordinates with respect to the chart.*

Definition B.3 (Smoothly Compatible Charts). *Let (U, φ) and (V, ψ) be two charts, they are said to be smoothly compatible, if they are either disjoint, or both the transition map $\psi \circ \varphi^{-1} : \varphi(U \cap V) \to \psi(U \cap V)$ and its inverse $\varphi \circ \psi^{-1}$ are infinitely differentiable.*

Definition B.4 (Smooth Atlas). *A smooth atlas is a collection of charts covering the manifold which are smoothly compatible with each other. The smooth atlas is said to be maximal if every chart that is smoothly compatible of each chart in the atlas has been in the atlas.*

Definition B.5 (Smooth Manifold). *A smooth manifold is a topological n-manifold M with a maximal smooth atlas $\mathcal{A}$. This smooth atlas is also called the smooth structure of M.*

B.2 Smooth Maps

Definition B.6 (Smooth Map). *Let M and N be smooth manifolds, then $F : M \to N$ is said to be smooth if for every $p \in M$, there exists local charts (U, φ) and (V, ψ) which respectively contain p and $F(p)$, such that the function $\hat{F} = \psi \circ F \circ \varphi^{-1}$ is infinitely differentiable from $\varphi(U)$ to $\psi(V)$. Here, the function $\hat{F}$ is called the coordinate representation of F with respect to the given coordinate charts.*

Especially, the function $F : M \to \mathbb{R}^k$ is said to be *smooth*, if for each $p \in M$, there exists a local chart (U, φ) that contains p such that $\hat{f} = f \circ \varphi^{-1}$ is infinitely differentiable.

The set of all smooth real-valued functions $f : M \to \mathbb{R}$ constitutes a real vector space, denoted by $C^\infty(M)$.

Smooth Maps have the following properties

1. *Smoothness is local.* It means that a map $F : M \to N$ is smooth, if and only if for every point $p \in M$, there exists a neighborhood of p, say U, such that the restriction $F|_U$ is smooth.

2. Let M and N be smooth manifolds, and let $F : M \to N$ be continuous maps. $\{(U_\alpha, \varphi_\alpha\}$ and $\{V_\beta, \psi_\beta\}$ are smooth atlases for M and N. Then F is smooth if and only if for each α and β with $U_\alpha \cap V_\beta \neq \emptyset$, $\psi_\beta^{-1} \circ F \circ \varphi_\alpha$ is smooth on its domain of definition.

3. Any composition of smooth maps between smooth manifolds is smooth.

Definition B.7 (Diffeomorphism). *A diffeomorphism between smooth manifolds M and N is a smooth bijective map, whose inverse is also smooth. Two manifolds are said to be diffeomorphic is there exists a diffeomorphism between them.*

B.3 Tangent Vectors and Tangent Space

Tangent Space is a local linear approximation of the manifold, which is the basis of Lie algebra theory. The elements in a tangent space are tangent vectors, which have close relations with derivatives of smooth curves.

Definition B.8 (Derivation). *Let M be a smooth manifold and $p \in M$. A linear map $X : C^\infty(M) \to \mathbb{R}$ is called a derivation at p if it satisfies*

$$X(fg) = f(p)Xg + g(p)Xf,$$

for all $f, g \in C^\infty(M)$.

Definition B.9 (Tangent Space). *The set of all derivations of $C^\infty(M)$ at p constitute a vector space, which is called the tangent space of M at p, denoted by T_pM. Each element in T_pM is called a tangent vector.*

We can see that each tangent vector corresponds to a derivation functional.

Tangent space of an n-dimensional manifold is isomorphic to $\mathbb{R}^n$.

Push-forwards are linear maps between tangent spaces.

Definition B.10 (Push-forward). *Let M and N be smooth manifolds, and $F : M \to$*

N be a smooth map, for each $p \in M$, the map $F_* : T_pM \to T_{F(p)}N$ defined as

$$(F_*X)(f) = X(f \circ F)$$

for each $f \in C^\infty(N)$, is called the push-forward associated with F.

It can be easily shown that $F_* : T_pM \to T_{F(p)}N$ is linear.

In addition, the map that takes F to F_* has the following properties:

1. It takes identity map to identity map: $(\mathrm{Id}_M)_* = \mathrm{Id}_{T_pM}$.
2. $(G \circ F)_* = G_* \circ F_*$.
3. If F is a diffeomorphism, then F_* is an isomorphism.

Let (U, φ) be a smooth chart on an n-manifold M, then φ is a diffeomorphism between U and its range, hence, $\varphi_* : T_pM \to T_{\varphi(p)}\mathbb{R}^n$ is an isomorphism, so as $(\varphi^{-1})_*$.

$T_{\varphi(p)}\mathbb{R}^n$ has a natural basis with the derivations $\partial/\partial x^i|_{\varphi(p)}, i = 1, 2, \ldots, n$. Then $(\varphi^{-1})_*$ will take this basis to form a basis of T_pM:

$$\left.\frac{\partial}{\partial x^i}\right|_p = (\varphi^{-1})_* \left.\frac{\partial}{\partial x^i}\right|_{\varphi(p)},$$

which acts on $f \in C^\infty(M)$ as

$$\left.\frac{\partial}{\partial x^i}\right|_p f = \left.\frac{\partial}{\partial x^i}\right|_{\varphi(p)} (f \circ \varphi^{-1}) = \frac{\partial \hat{f}}{\partial x^i}(\hat{p}),$$

where $\hat{f}$ and $\hat{p}$ are respectively the coordinate representations of f and p with respect to the given chart.

Then any vector $X \in T_pM$ can be uniquely written as a linear combination of the basis

$$X = \sum_i X^i \left.\frac{\partial}{\partial x^i}\right|_p,$$

which acts at a smooth function $f \in C^\infty(M)$ as

$$Xf = \sum_i X^i \frac{\partial \hat{f}}{\partial x^i}(\hat{p}).$$

Here $(X^1, X^2, \ldots, X^n)$ is the coordinate representation of X, which can be obtained by

$$X^i = X(x^i),$$

where x^i is the i-th coordinate function in C^∞.

Consider a smooth map $F : U \to V$ with $U \in \mathbb{R}^n$ and $V \in \mathbb{R}^m$, then F_* is a linear operator from an n-dimensional space to an m-dimensional space, which thus can be represented by an $m \times n$ matrix. It can be shown that, this matrix is exactly the *Jacobian matrix* of F at p, denoted by $DF(p)$.

For a smooth map $F : M \to N$ between two general manifolds, the matrix representation can be obtained with respect to fixed charts.

Definition B.11 (Curve). *Let M be a manifold, a curve in M is a continuous map $\gamma : J \to M$, with $J \in \mathbb{R}$ being an interval. If M is a smooth manifold, and γ is a smooth map, it is called a smooth curve.*

Definition B.12 (Tangent Vector to a curve). *Let γ be a smooth curve in a smooth manifold M, the tangent vector to γ at $t_0 \in J$ is*

$$\gamma'(t_0) = \gamma_* \left(\left. \frac{d}{dt} \right|_{t_0} \right) \in T_{\gamma(t_0)} M.$$

It acts on a function $f \in C^\infty M$ as a derivation by

$$\gamma'(t_0) f = \left. \frac{d}{dt} \right|_{t_0} (f \circ \gamma) = \frac{d(f \circ \gamma)}{dt}(t_0).$$

Let (U, φ) be a smooth chart on M that contains $\gamma(t_0)$, then we can have

$$\gamma'(t_0) f = (\gamma^i)'(t_0) \left. \frac{\partial}{\partial x^i} \right|_{\gamma(t_0)},$$

where $\gamma^i(t)$ is the i-th coordinate component of $\gamma(t)$, $((\gamma^1)'(t_0), \ldots, (\gamma^n)'(t_0))$ is the coordinate representation.

There exists close relations between tangent space and smooth curves.

Theorem B.1. *Let M be a smooth manifold and $p \in M$. Each $X \in T_pM$ is the tangent vector to some smooth curve in M.*

For composite curve, we have

Let $F : M \to N$ be a smooth map, and $\gamma : J \to M$ be a smooth curve in M, then $F \circ \gamma : J \to N$ is a smooth curve in N. The tangent vector at $t = t_0$ to the composite curve $F \circ \gamma$ is given by

$$(F \circ \gamma)'(t_0) = F_*(\gamma'(t_0)). \tag{B.1}$$

This proposition is often utilized to compute the push-forwards, as

$$F_*X = (F \circ \gamma)'(0), \tag{B.2}$$

where γ is some smooth curve whose tangent vector is X at $t = 0$.

B.4 Vector Fields

Definition B.13 (Tangent bundle)**.** *The tangent bundle of a smooth manifold M, denoted by TM, is the disjoint union of the tangent spaces at all points of M:*

$$TM = \bigsqcup_{p \in M} T_pM.$$

Each element in TM is a pair (p, X) with $X \in T_pM$.

The tangle bundle has a *natural projection map* $\pi : TM \to M$, which maps (p, X) to p. The natural projection map is a smooth map.

Definition B.14 (Vector Field)**.** *A vector field is a continuous map $Y : M \to TM$,*

usually written as $p \mapsto Y_p$, *which satisfies*

$$\pi \circ Y = \mathrm{Id}_M.$$

In other words, it is a continuous map that maps each point p *in* M *to a tangent vector in the corresponding tangent space* T_pM.

If the map is smooth, it is called a smooth vector field.

Considering that each tangent vector is a derivation operator, then each vector field is also a map that maps a real-valued smooth function to a real-valued smooth function, by taking derivatives at each point in M using the corresponding tangent vector.

B.5 Embedding and Submanifolds

Definition B.15 (Rank of Smooth Map). *Let* M *and* N *be smooth manifolds, and* $F : M \to N$ *be a smooth map. The rank of* F *at* $p \in M$ *is the rank of the linear map* $F_* : T_pM \to T_{F(p)}N$, *which is just the rank of the Jacobian matrix* $DF(p)$ *with respect to any smooth chart.*

If a smooth map F *has the same rank* k *at every* $p \in M$, *it is said to have a constant rank* k, *denoted by* $\mathrm{rank}(F) = k$.

Definition B.16 (Submersion). *A smooth map* $F : M \to N$ *is called a submersion if* F_* *is surjective at every point, i.e.* $\mathrm{rank}(F) = \dim N$.

Definition B.17 (Immersion). *A smooth map* $F : M \to N$ *is called an immersion if* F_* *is injective at every point, i.e.* $\mathrm{rank}(F) = \dim M$.

Definition B.18 (Smooth Embedding). *A smooth embedding is an immersion* $F : M \to N$ *that is also a topological embedding, i.e. a homeomorphism onto its image* $F(M) \subset N$.

The inverse function theorem relates the rank of a smooth map to invertibility.

Theorem B.2 (Inverse Function Theorem). *Let M and N be smooth manifolds, $p \in M$, and $F : M \to N$ is a smooth map such that $F_* : T_pM \to T_{F(p)}N$ is bijective, i.e. $DF(p)$ is of full rank, then there exists connected neighborhoods U_0 of p and V_0 of $F(p)$ such that $F|_{U_0} : U_0 \to V_0$ is a diffeomorphism.*

As a consequence, we have the following theorem.

Theorem B.3 (Rank Theorem). *Suppose M and N are two smooth manifolds, and $F : M \to N$ is a smooth map with constant rank k. For each $p \in M$, there exist smooth coordinate charts (U, φ) centered at p, and (V, ψ) such that*

$$\psi \circ F \circ \varphi^{-1}(x^1, \ldots, x^k, x^{k+1}, \ldots, x^m) = (x^1, \ldots, x^k, 0, \ldots, 0).$$

Smooth submanifolds are modeled locally as embedding of $\mathbb{R}^k$ into $\mathbb{R}^n$, identifying $\mathbb{R}^k$ with the subspace of $\mathbb{R}^n$ in form of

$$\{(x^1, \ldots, x^k, x^{k+1}, \ldots, x^n) | x^{k+1} = \cdots = x^n = 0\}.$$

Definition B.19 (k-slice). *If U is an open subset of $\mathbb{R}^n$, a k-slice of U is any subset in the form of*

$$S = \{(x^1, \ldots, x^k, x^{k+1}, \ldots, x^n) | x^{k+1} = c^{k+1}, \ldots, x^n = c^n\},$$

for some constants $c^{k+1}, \ldots, c^n$.

The definition can be easily extended to generic manifold. Let M be a smooth n-manifold, and (U, φ) be a smooth chart on M. A subset $S \subset U$ is a k-slice of U if $\varphi(S)$ is a k-slice of $\varphi(U)$.

Definition B.20 (Embedded Submanifold). *A subset $S \subset M$ is called an embedded submanifold of dimension k, or embedded k-submanifold, if for each point $p \in S$, there exists a smooth chart (U, φ) on M such that $p \in U$ and $U \cap S$ is a k-slice of U.*

Here, the chart (U, φ) is called a slice chart for S in M.

Let S be an embedded submanifold of M, then $\dim M - \dim S$ is called the *codimension* of S in M.

Embedded submanifolds and embedding have close relations.

Theorem B.4. *The image of a smooth embedding is an embedded submanifold.*

Conversely, let S be an embedded submanifold of M, it has a unique smooth structure such that the inclusion map $S \hookrightarrow M$ is a smooth embedding.

The tangent space to a submanifold is a subspace of the tangent space to the ambient manifold.

Theorem B.5. *Suppose $S \subset M$ is an embedded submanifold and $p \in S$. The tangent space T_pS is a subspace of T_pM, and it is given by*

$$T_pS = \{X \in T_pM : Xf = 0 \text{ whenever } f \in C^\infty(M) \text{ and } f|_S \equiv 0\}$$

Appendix C

Affine Transformation Group

In geometry, an *affine transform* is a transformation which preserves straight lines and ratios of distances between points lying on a straightline. Affine transforms are one of the most important family of geometric transforms and have been extensively studied. In this work, we derive the first family of geometric flows – the *affine flows* by extending affine transforms into continuous transform processes. Moreover, one can construct more complex flows, using affine flows as the basic building blocks.

This section provides a brief review of affine transforms, and discusses their Lie algebraic representations.

C.1 The Affine Transformation Group

The affine transform can be defined in either an algebraic way or a geometric way.

Definition C.1 (Affine Transformation). *In algebra, an affine transformation, or called an affine map, between two vector spaces consists of a linear transformation followed by a translation*

$$T\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{t},$$

here, the affine transformation is characterized by the pair $(\mathbf{A}, \mathbf{t})$*, where* $\mathbf{A}$ *represents the linear transformation, while* $\mathbf{t}$ *represents the translation.*

In geometry, an affine transformation is a map that preserves collinearity of points

and ratios of distances between collinear points.

Affine transformation is not always invertible. An affine transformation $(\mathbf{A}, \mathbf{t})$ is invertible if and only if the linear map $\mathbf{A}$ is invertible. In this notes, we only discuss invertible affine transformations. Without explicit statement, an affine transformation means an invertible one in the following text.

Definition C.2 (Affine Transformation Group). *The set of all invertible affine transformations for n-dimensional space form the affine transformation group, or called Affine Group, denoted by* $\mathrm{Aff}(n)$*, with the group operations defined as follows. Let* $T_1 = (\mathbf{A}_1, \mathbf{t}_1)$ *and* $T_2 = (\mathbf{A}_2, \mathbf{t}_2)$ *be two affine transforms, their composition acts on* $\mathbf{x}$ *as*

$$(T_1 \circ T_2)(\mathbf{x}) = T_1(T_2(\mathbf{x})) = \mathbf{A}_1\mathbf{A}_2\mathbf{x} + (\mathbf{t}_1 + \mathbf{A}_1\mathbf{t}_2).$$

Hence, the multiplication is defined as

$$(\mathbf{A}_1, \mathbf{t}_1) \cdot (\mathbf{A}_2, \mathbf{t}_2) = (\mathbf{A}_1\mathbf{A}_2,\ \mathbf{t}_1 + \mathbf{A}_1\mathbf{t}_2).$$

In addition, the inverse of $T = (\mathbf{A}, \mathbf{t})$ *is*

$$T^{-1} = (\mathbf{A}^{-1}, -\mathbf{A}^{-1}\mathbf{t}).$$

The identity element of the group $\mathrm{Aff}(n)$ *is the identity transform* $(\mathbf{I}, \mathbf{0})$.

It can be shown that the affine transformation group is a Lie group with the natural topology of the Cartesian product space $\mathrm{GL}(n, \mathbb{R}) \times \mathbb{R}^n$. The affine transformation group is not commutative.

Homogeneous coordinates and Matrix representation

Next, we discuss how an affine transform can be represented in a matrix form. An affine transformation is generally not linear, and thus it cannot be directly represented by a matrix.

To derive the matrix representation of affine transforms, *homogeneous coordinates* is introduced.

Definition C.3 (Homogeneous Coordinates). *In an n-dimensional space, the homogeneous coordinate of a point* $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ *is a vector of length* $n+1$*, which augments* $\mathbf{x}$ *with an extra one. In this notes, it is denoted by* $\tilde{\mathbf{x}}$*, as*

$$\tilde{\mathbf{x}} = (x_1, x_2, \ldots, x_n, 1)^T.$$

Using homogeneous coordinates, the affine transformation defined above can be written as

$$\begin{bmatrix} T\mathbf{x} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

Hence, the affine transformation $(\mathbf{A}, \mathbf{t})$ can be uniquely represented by the $(n+1) \times (n+1)$ matrix $\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$. The matrix is invertible if and only if the transformation is invertible.

With the matrix representation, we can formulate the application and composition of affine transforms in form of matrix multiplication. What's more important, the matrices in the aforementioned form constitute a group, which is a Lie subgroup of $\mathrm{GL}(n+1, \mathbb{R})$. The affine transformation group $\mathrm{Aff}(n)$ is isomorphic to this matrix Lie group. In the following discussion, we treat them as identical.

Lie algebraic representation

According to the theory of Lie group and Lie algebra, for each matrix Lie group, there exists a corresponding Lie algebra. The Lie algebra of $\mathrm{Aff}(n)$, denoted by $\mathfrak{aff}(n)$, comprises all the matrices such that the induced one-parameter subgroups lie in $\mathrm{Aff}(n)$, that is

$$\mathbf{X} \in \mathfrak{aff}(n) \quad \Leftrightarrow \quad \forall t \in \mathbb{R},\ e^{t\mathbf{X}} \in \mathrm{Aff}(n).$$

Hence, $\frac{d}{dt} e^{t\mathbf{X}}|_{t=0} = \mathbf{X}$. As the bottom row of the matrix representation of affine transforms are fixed, the bottom row of each matrix $\mathbf{X} \in \mathfrak{aff}(n)$ must be an zero row

vector. In addition, for each matrix in form of

$$\mathbf{X} = \begin{bmatrix} \mathbf{Y} & \mathbf{u} \\ \mathbf{0} & 0 \end{bmatrix},$$

we have

$$e^{\mathbf{X}} = \exp\left(\begin{bmatrix} \mathbf{Y} & \mathbf{u} \\ \mathbf{0} & 0 \end{bmatrix}\right) = \begin{bmatrix} e^{\mathbf{Y}} & \varphi(\mathbf{Y})\mathbf{u} \\ \mathbf{0} & 1 \end{bmatrix} \tag{C.1}$$

Here, we have

$$\varphi(\mathbf{Y}) = \sum_{k=1}^{\infty} \frac{\mathbf{Y}^{k-1}}{k!},$$

which converges for every $\mathbf{Y}$. When $\mathbf{Y}$ is invertible,

$$\varphi(\mathbf{Y}) = (e^{\mathbf{Y}} - \mathbf{I})\mathbf{Y}^{-1} = \mathbf{Y}^{-1}(e^{\mathbf{Y}} - \mathbf{I}).$$

Therefore, $e^{t\mathbf{X}} \in \mathrm{Aff}(n)$. Hence, the Lie algebra $\mathfrak{aff}(n)$ of $\mathrm{Aff}(n)$ consists of all $(n+1) \times (n+1)$ consists of every matrix with zero bottom row. The dimensionality of the Lie algebra $\mathfrak{aff}(n)$ is equal to that of the Lie group $\mathrm{Aff}(n)$, which is

$$\dim \mathrm{Aff}(n) = \dim \mathfrak{aff}(n) = n(n+1). \tag{C.2}$$

In particular, when $n = 2$ (*i.e.* for a two-dimensional space), the dimension is 6.

C.2 Factorization of the Affine Group

Next, we perform a complete analysis of the affine transform group, in order to obtain a deeper understanding of its structure. The process would lead to a series of important subgroups with interesting geometric and algebraic characteristics.

General linear group and translation group

First of all, the affine group $\mathrm{Aff}(n)$ can be factorized as the semidirect product between the *general linear group* $\mathrm{GL}(n, \mathbb{R})$ and the *translation group*, which is isomorphic to

$(\mathbb{R}^n, +)$, as

$$\mathrm{Aff}(n) = \mathrm{GL}(n, \mathbb{R}) \ltimes \mathbb{R}^n. \tag{C.3}$$

This equations suggests that *each affine transformation is uniquely expressed as a linear transform followed by a translation.*

Particularly, the *translation group* consists of all translation transforms. Each translation is characterized by a translation vector $\mathbf{t}$, and acts on a point $\mathbf{x}$ as

$$T\mathbf{x} = \mathbf{x} + \mathbf{t}.$$

The group product of translations $\mathbf{t}_1$ and $\mathbf{t}_2$ is $\mathbf{t}_1 + \mathbf{t}_2$, and the inverse of the translation $\mathbf{t}$ is $-\mathbf{t}$. The identity element is given by $\mathbf{0}$. The translation group of an n-dimensional vector space is an n-dimensional Lie group, which is commutative and isomorphic to the additive group of real vector space $(\mathbb{R}^n, +)$.

The *general linear group*, denoted by $\mathrm{GL}(n, \mathbb{R})$, consists of all invertible real matrices. $\mathrm{GL}(n, \mathbb{R})$ is an n^2-dimensional Lie group. As a matrix Lie group, its Lie algebra, denoted by $\mathfrak{gl}(n, \mathbb{R})$, consists of all $n \times n$ matrices.

$\mathrm{GL}(n, \mathbb{R})$ is not connected, and comprises two components. The matrices in one component have positive determinant, while those in the other component have negative determinant. The component of matrices with positive determinants in itself is a Lie subgroup, denoted by $\mathrm{GL}^+(n, \mathbb{R})$.

Specifically, given a particular axis $\mathbf{b}$, the linear group $\mathrm{GL}(n, \mathbb{R})$ can be factorized into a semidirect product as

$$\mathrm{GL}(n, \mathbb{R}) = \mathrm{GL}^+(n, \mathbb{R}) \rtimes (\{1, -1\}, \times), \tag{C.4}$$

where $(\{1, -1\}, \times)$ is isomorphic to the reflection group, with 1 corresponding to the identity map, while -1 corresponding to the reflection along the axis $\mathbf{b}$. The above equation essentially establishes an important fact: *Given an axis, each linear transform can be either a transform with positive determinant or a transform with positive determinant followed by a reflection along the given axis.*

Special linear group and uniform scaling

Consider the determinant function det : $\mathrm{GL}^+(n,\mathbb{R}) \to \mathbb{R}^+$ that maps each linear transform matrix to its determinant value. It is not difficult to see that this function is a group homomorphism. The kernel of the determinant function is what we call the *special linear group*, while the induced quotient group is isomorphic to the multiplicative group of positive real numbers $(\mathbb{R}^+, \times)$, which is also isomorphic to the group of *uniform scaling*.

This suggests that $\mathrm{GL}^+(n,\mathbb{R})$ can be further decomposed into a direct product of a *special linear group* $\mathrm{SL}(n,\mathbb{R})$ and a the group of uniform scaling, as

$$\mathrm{GL}^+(n,\mathbb{R}) = \mathrm{SL}(n,\mathbb{R}) \times (\mathbb{R}^+, \times) \tag{C.5}$$

The *special linear group* over the real number field, denoted by $\mathrm{SL}(n,\mathbb{R})$, consists of all linear transforms with determinant one. All transforms in $\mathrm{SL}(n,\mathbb{R})$ are volume-preserving. The dimension of $\mathrm{SL}(n,\mathbb{R})$ is $n^2 - 1$.

The *uniform scaling group* consists of all *uniform scaling* transforms. Each uniform scaling is characterized by a scaling factor $s \in \mathbb{R}^+$, and acts on a point $\mathbf{x}$ as

$$T\mathbf{x} = s\mathbf{x}.$$

The uniform scaling group is a one-dimensional Abelian group, which is naturally isomorphic to the multiplicative group of positive real numbers. It is the normal subgroup of $\mathrm{GL}^+(n,\mathbb{R})$ and $\mathrm{GL}(n,\mathbb{R})$.

Orthogonal group and special orthogonal group

A subgroup of the general linear group deserves special attention, that is, the group of transforms that preserve Euclidean distances, called *orthogonal group*. The orthogonal group of n-dimensional space, denoted by $\mathrm{O}(n)$, is the group of all $n \times n$ orthogonal matrices. A matrix $\mathbf{A}$ is said to be orthogonal, if $\mathbf{A}^{-1} = \mathbf{A}^T$, i.e. $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}$.

The dimension of $\mathrm{O}(n)$ is $n(n-1)/2$. The determinant of an orthogonal matrix

can be either 1 or -1. Correspondingly, the orthogonal group has two connected components, respectively contain the matrices of determinant 1 and -1. The component corresponding to the determinant one is a Lie subgroup of $\mathrm{O}(n)$ of the same dimension, called *special orthogonal group*, denoted by $\mathrm{SO}(n)$. In general, $\mathrm{SO}(n)$ is not commutative, except when $n = 1$ (in this case, it degenerates to the trivial group 1) and $n = 2$ (the 2D rotation group).

The transforms in $\mathrm{O}(n)$ preserve inner product, as

$$\forall \mathbf{R} \in \mathrm{O}(n), \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \ \langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{Rx}, \mathbf{Ry} \rangle.$$

As an immediate corollary, they preserve Euclidean distance, as

$$\forall \mathbf{R} \in \mathrm{O}(n), \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \ ||\mathbf{x} - \mathbf{y}|| = ||\mathbf{Rx} - \mathbf{Ry}||.$$

Conversely, *every linear transform that preserves distances is orthogonal.* In this sense, the orthogonal group can be defined as the group of all distance-preserving linear transforms.

Euclidean group

The transforms in translation group and orthogonal group, as well as their compositions are all distance-preserving, which together with their composite transforms constitute the *Euclidean group.*

The *Euclidean group* of an n-dimensional vector space, denoted by $\mathrm{E}(n)$, consists of all distance-preserving affine transforms. Each affine transform in $\mathrm{E}(n)$ is given by $(\mathbf{R}, \mathbf{t})$, in which $\mathbf{R} \in \mathrm{O}(n)$. The dimension of $\mathrm{E}(n)$ is $n(n+1)/2$. It can be factorized as the semidirect product of the orthogonal group and the translation group.

$$\mathrm{E}(n) = \mathrm{O}(n) \ltimes \mathbb{R}^n. \tag{C.6}$$

In other words, each Euclidean transform can be uniquely expressed by an orthogonal transform followed by a translation. Like $\mathrm{GL}(n, \mathbb{R})$, $\mathrm{E}(n)$ has two connected compo-

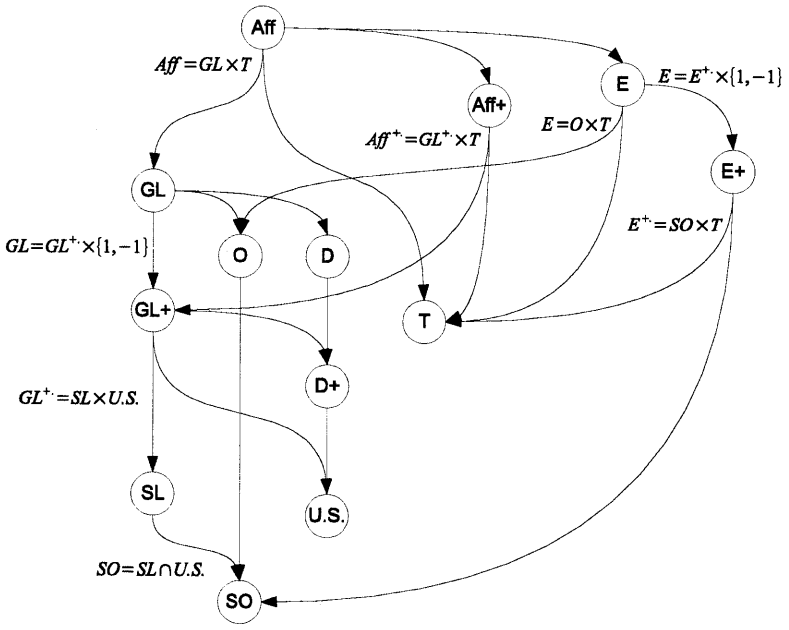


Figure C-1: This graph illustrates the relations of subgroups of the Affine group. In this graph, `Aff` represents $\mathrm{Aff}(n)$, `Aff+` represents $\mathrm{Aff}^+(n)$, `GL` represents $\mathrm{GL}(n,\mathbb{R})$, `GL+` represents $\mathrm{GL}^+(n,\mathbb{R})$, `O` represents $\mathrm{O}(n)$, `SO` represents $\mathrm{SO}(n)$, `D` represents the diagonal group, `D+` represents the positive diagonal group, `U.S.` represents the uniform scaling group, `T` represents the translation group, `E` represents $\mathrm{E}(n)$, `E+` represents $\mathrm{E}^+(n)$. The arrows represent the sub-group relationship, while the symbol $\times$ in the formulas represents the semidirect product factorization.

nents, respectively corresponding to determinants 1 and -1. The component with transforms of determinant one in itself is a group, denoted by $\mathrm{E}^+(n)$, which has

$$\mathrm{E}^+(n) = \mathrm{SO}(n) \ltimes \mathbb{R}^n. \tag{C.7}$$

Figure C-1 shows the relations between the aforementioned subgroups and how the affine group is factorized into semidirect product of subgroups.

C.3 Two-dimensional Affine Transforms

In this work, we particularly focuses on the geometric transforms on a two dimensional space. Here, a detailed analysis is performed on the two-dimensional affine group and its associated Lie algebra.

In a two-dimensional space, an affine transform is characterized by a 2×2 matrix $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and a translation vector $\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$. Let the transform $T = (\mathbf{A}, \mathbf{t})$ send a point $\mathbf{x} = (x_1, x_2)$ to $\mathbf{x}' = (x_1', x_2')$, we have

$$\begin{cases} x_1' = a_{11}x_1 + a_{12}x_2 + t_1, \\ x_2' = a_{21}x_1 + a_{22}x_2 + t_2. \end{cases}$$

All invertible 2D affine transforms constitute the 2D affine group, denoted Aff(2), of dimension 6.

The Lie algebra of Aff(2), denoted $\mathfrak{aff}(2)$, consists of all 3×3 matrices in the following form

$$\begin{bmatrix} \mathbf{Y} & \mathbf{u} \\ \mathbf{0} & 0 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & u_1 \\ y_{21} & y_{22} & u_2 \\ 0 & 0 & 0 \end{bmatrix}.$$

Evaluation of the exponentiation mapping

Let

$$\exp\left(\begin{bmatrix}\mathbf{Y} & \mathbf{u}\\ \mathbf{0} & 0\end{bmatrix}\right) = \begin{bmatrix}\mathbf{A} & \mathbf{t}\\ \mathbf{0} & 1\end{bmatrix}, \tag{C.8}$$

then the matrices are related as follows.

$$\mathbf{A} = e^{\mathbf{Y}} \quad \text{and} \quad \mathbf{t} = \varphi(\mathbf{Y})\mathbf{u} = \sum_{k=1}^{\infty} \frac{\mathbf{Y}^{k-1}}{k!}\mathbf{u}. \tag{C.9}$$

In this above, $\varphi(\mathbf{Y})$ is related to $e^{\mathbf{Y}}$ as

$$e^{\mathbf{Y}} = \mathbf{I} + \varphi(\mathbf{Y})\mathbf{Y} = \mathbf{I} + \mathbf{Y}\varphi(\mathbf{Y}). \tag{C.10}$$

To derive the method of evaluating $\varphi(\mathbf{Y})$, we first generalize $\varphi(\cdot)$, getting a family of functions $\{\varphi_{(l)} : M_n(\mathbb{R}) \to \mathrm{GL}(n, \mathbb{R})\}_l$ indexed by a nonnegative integer l as

$$\varphi_{(l)}(\mathbf{Y}) = \sum_{k=l}^{\infty} \frac{\mathbf{Y}^{k-l}}{k!} \tag{C.11}$$

It can be shown that for each l, this series is convergent. Obviously, when $l = 0$, it is the exponential mapping, when $l = 1$, it is $\varphi(\cdot)$ defined above. Moreover, the identity in (C.10) can be generalized to

$$\varphi_{(l)}(\mathbf{Y}) = \frac{1}{l!}\mathbf{I} + \varphi_{(l+1)}(\mathbf{Y})\mathbf{Y} = \frac{1}{l!}\mathbf{I} + \mathbf{Y}\varphi_{(l+1)}(\mathbf{Y}) \tag{C.12}$$

Especially, we have

$$\varphi(\mathbf{Y}) = \mathbf{I} + \varphi_{(2)}(\mathbf{Y})\mathbf{Y} = \mathbf{I} + \mathbf{Y}\varphi_{(2)}(\mathbf{Y}). \tag{C.13}$$

When $\mathbf{Y}$ is a 2×2 matrix, the computation can be done as follows. If $\mathbf{Y}$ is non-singular, then

$$\varphi(\mathbf{Y}) = \mathbf{Y}^{-1}(e^{\mathbf{Y}} - \mathbf{I}) = (e^{\mathbf{Y}} - \mathbf{I})\mathbf{Y}^{-1}. \tag{C.14}$$

When $\mathbf{Y}$ is singular, we can accomplish the computation as below

$$\varphi(\mathbf{Y}) = \mathbf{I} + \varphi_{(2)}\left(\mathrm{tr}(\mathbf{Y})\right)\mathbf{Y}. \tag{C.15}$$

with

$$\varphi_{(2)}(x) = \begin{cases} x^{-2}(e^x - 1 - x), & x \neq 0 \\ 0.5, & x = 0. \end{cases}. \tag{C.16}$$

C.4 Entries of the Lie algebraic representation

Here, we are going to investigate the meaning of each entry in the Lie algebraic representation and study how they are related to the transforms. For the affine transform $\mathbf{T} = e^{\mathbf{X}}$, we write the Lie algebraic representation $\mathbf{X}$ as

$$\mathbf{X} = \begin{bmatrix} \mathbf{Y} & \mathbf{u} \\ \mathbf{0} & 0 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & u_1 \\ y_{21} & y_{22} & u_2 \\ 0 & 0 & 0 \end{bmatrix}$$

.

Let $\mathbf{E}_{ij}$ be a 3×3 matrix with the entry at the i-th row and j-th column being 1 and others being 0. Then, the canonical decomposition of $\mathbf{X}$ can be given by

$$\mathbf{X} = y_{11}\mathbf{E}_{11} + y_{12}\mathbf{E}_{12} + y_{21}\mathbf{E}_{21} + y_{22}\mathbf{E}_{22} + u_1\mathbf{E}_{13} + u_2\mathbf{E}_{23}. \tag{C.17}$$

Now, we examine these six components respectively.

Logarithm of scaling factors: y_{11} and y_{22}

$$\exp(y_{11}\mathbf{E}_{11}) = \begin{bmatrix} e^{y_{11}} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} : \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} e^{y_{11}}x_1 \\ x_2 \end{bmatrix}$$

$$\exp(y_{22}\mathbf{E}_{22}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{y_{22}} & 0 \\ 0 & 0 & 1 \end{bmatrix} : \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ e^{y_{22}} x_2 \end{bmatrix}$$

Hence, the transforms corresponding to $y_{11}\mathbf{E}_{11}$ and $y_{22}\mathbf{E}_{22}$ are the scalings respectively along x-axis and y-axis, by factors $e^{y_{11}}$ and $e^{y_{22}}$. In this sense, y_{11} and y_{22} encode the logarithm of scale factors along x-axis and y-axis.

Shearing coefficients: y_{12} and y_{21}

$$\exp(y_{12}\mathbf{E}_{12}) = \begin{bmatrix} 1 & y_{12} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} : \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 + y_{12}x_2 \\ x_2 \end{bmatrix}$$

$$\exp(y_{21}\mathbf{E}_{21}) = \begin{bmatrix} 1 & 0 & 0 \\ y_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} : \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_2 + y_{21}x_1 \end{bmatrix}$$

Hence, the transforms corresponding to $y_{12}\mathbf{E}_{12}$ and $y_{21}\mathbf{E}_{21}$ are the shearing transforms respectively parallel to x-axis and y-axis, with coefficients y_{12} and y_{21}. In this sense, y_{12} and y_{21} encode the shearing coefficients parallel to x-axis and y-axis.

Translation displacement: u_1 and u_2

$$\exp(u_1\mathbf{E}_{13}) = \begin{bmatrix} 1 & 0 & u_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} : \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 + u_1 \\ x_2 \end{bmatrix}$$

$$\exp(u_2\mathbf{E}_{23}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{bmatrix} : \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_2 + u_2 \end{bmatrix}$$

Hence, the transforms corresponding to $u_1\mathbf{E}_{13}$ and $u_2\mathbf{E}_{23}$ are translations along x-direction and y-direction respectively, and u_1, u_2 encode the amount of displacement.

When $\mathbf{X}$ is a combination of multiple components, transforms of different types will be interleaved together to form the compound transform. In this process, the meaning of the individual elements will not be exactly what has been described above.

C.5 Important Subgroups of the 2D Affine Group

Now, we study the algebraic characteristics of several important geometric transforms.

Translation

Translation is moving points along some direction by a constant distance. It is characterized by the translation vector $\mathbf{t} = (t_1, t_2)^T$, and can be expressed by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + t_1 \\ x_2 + t_2 \end{pmatrix}$$

Here, t_1 and t_2 respectively represent the displacement along two axes. All translations form a subgroup of Aff(2), in which, the composition of two translations $\mathbf{t}_1$ and $\mathbf{t}_2$ is $\mathbf{t}_1 + \mathbf{t}_2$, and the inverse of the translation $\mathbf{t}$ is $-\mathbf{t}$. This group is an Abelian group isomorphic to $(\mathbb{R}^2, +)$.

The matrix representation of the translation $\mathbf{t}$ is $\mathbf{T} = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$. It is Lie algebraic representation is $\mathbf{X} = \begin{bmatrix} \mathbf{0} & \mathbf{t} \\ \mathbf{0} & 0 \end{bmatrix}$. $\mathbf{T}$ has a three-folded eigenvalue 1, and $\mathbf{X}$ has a three-folded eigenvalue 0. When $\mathbf{t} \neq \mathbf{0}$, both $\mathbf{T}$ and $\mathbf{X}$ are not diagonalizable, as the dimension of eigenspace is 2. Translation is isometric, and except for the identity map, translation has no invariant point.

Rotation

Rotation is moving points around the origin in a circular manner, keeping the distance from each point to the origin unchanged. It is characterized by the rotation radian

θ, and can be expressed by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 \cos\theta - x_2 \sin\theta \\ x_1 \sin\theta + x_2 \cos\theta \end{pmatrix}.$$

All rotations form a subgroup of $\mathrm{GL}(2, \mathbb{R})$, in which the composition of two rotations with radians θ_1 and θ_2 results in the rotation with radian $\theta_1 + \theta_2$, and the inverse of the rotation with radian θ is the one with radian $-\theta$. Its dimension is 1.

This group is actually the two-dimensional special orthogonal group $\mathrm{SL}(2, \mathbb{R})$. It is Abelian and is isomorphic to the circle group *i.e.* the group consisting of unitary complex numbers under multiplication.

The matrix representation of the rotation with radian θ is given by $(\mathbf{R}(\theta), \mathbf{0})$, with $\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$. The Lie algebraic representation of $\mathbf{R}(\theta)$ is $\mathbf{Y} = \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}$. The eigenvalues of $\mathbf{Y}$ are $\lambda_{+,-} = \pm i\theta$, corresponding to eigenvectors $[1, \mp i]^T$. Hence, the eigenvalues of $\mathbf{R}(\theta)$ are $e^{\pm i\theta} = \cos\theta \pm i \sin\theta$, with the same eigenvalues. Rotation is isometric, and has an invariant point at origin.

Scaling

Scaling is to enlarge or diminish an object, keeping the angle with respect to axes. It is characterized by the scaling coefficients along axes (s_1, s_2) where $s_1 > 0$ and $s_2 > 0$, and can be expressed by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} s_1 x_1 \\ s_2 x_2 \end{pmatrix}.$$

All scaling form a subgroup of $\mathrm{GL}(2, \mathbb{R})$, in which the composition of two scaling transforms (s_1, s_2) and (s_1', s_2') results in the scaling transform $(s_1 s_1', s_2 s_2')$. The inverse of scaling (s_1, s_2) is $(1/s_1, 1/s_2)$.

The scaling group is a 2-dimensional Abelian group, which is naturally isomorphic to $(\mathbb{R}^+, \times)^2$.

When $s_1 = s_2$, the scaling is called *uniform scaling*. All uniform scalings form a

subgroup of the scaling group of dimension 1. Obviously, it is also Abelian, and it is isomorphic to $(\mathbb{R}^+, \times)$.

The matrix representation of the scaling (s_1, s_2) is given by $(\mathbf{S}, \mathbf{0})$, with $\mathbf{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix}$, and its Lie algebraic representation is $\mathbf{Y} = \begin{bmatrix} \log s_1 & 0 \\ 0 & \log s_2 \end{bmatrix}$.

When $s_1 \neq s_2$, $\mathbf{S}$ has two non-zero eigenvalues s_1 and s_2 with eigenvectors $[1, 0]^T$ and $[0, 1]^T$ respectively. And $\mathbf{Y}$ has two eigenvalues $\log s_1$ and $\log s_2$ with the same eigenvectors. When $s_1 = s_2 = s$ (the case of *uniform scaling*), $\mathbf{S}$ has a two-folded eigenvalue s which corresponds to the eigenspace $\mathbb{R}^2$, and $\mathbf{Y}$ also has a two-folded eigenvalue $\log s$ with the eigenspace being $\mathbb{R}^2$.

Shearing

Shearing is fixing all points on one axis and shifting other points parallel to the axis by a distance proportional to their perpendicular distance from the axis. It is characterized by a shearing coefficient α.

The shearing parallel to x-axis with coefficient α is expressed by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + \alpha x_2 \\ x_2 \end{pmatrix}.$$

While the shearing parallel to y-axis is expressed by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 \\ x_2 + \alpha x_1 \end{pmatrix}.$$

All shearing parallel to the x-axis form a subgroup of $\mathrm{GL}(2, \mathbb{R})$, in which the composition of shearing transforms with coefficients α_1 and α_2 results in the shearing with coefficient $\alpha_1 + \alpha_2$. The inverse of the shearing of coefficient α is that of coefficient $-\alpha$. It is a one-dimensional Abelian group, isomorphic to $(\mathbb{R}, +)$.

The matrix representation of the shearing along x-axis with coefficient α is given by $(\mathbf{A}, \mathbf{0})$, with $\mathbf{A} = \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix}$. Its Lie Algebraic representation is $\mathbf{Y} = \begin{bmatrix} 0 & \alpha \\ 0 & 0 \end{bmatrix}$.

$\mathbf{A}$ has a three-folded eigenvalue 1, while $\mathbf{Y}$ has a three-folded eigenvalue 0. When $\alpha \neq 0$, both $\mathbf{A}$ and $\mathbf{Y}$ are not diagonalizable, as the dimension of the eigenspace is 1 with basis $[1, 0]^T$. The shearing parallel to the y-axis has similar properties.

To sum up, all subgroups discussed above are Abelian groups. Except for scaling, others are volume-preserving. The translations and rotations are further distance-preserving. Actually, the Euclidean group in 2D space with positive determinant $\mathrm{E}^+(2)$ is the semidirect product of the rotation group and the translation group.

Bibliography

[1] Amr Ahmed and Eric Xing. Dynamic Non-Parametric Mixture Models and The Recurrent Chinese Restaurant Process : with Applications to Evolutionary Clustering. In *Proc. of SDM'08*, 2008.

[2] S. Ali and M. Shah. A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *Proc. of CVPR'07*, 2007.

[3] Saad Ali and Mubarak Shah. A supervised learning framework for generic object detection in images. In *Proc. of ICCV'05*, 2005.

[4] T. Amiaz, S. Fazekas, D. Chetverikov, and N. Kiryati. Detecting regions of dynamic textures. In *Conference on Scale Space and Variational Methods in Computer Vision*, 2007.

[5] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Countour detection and hierarchical image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(5), 2011.

[6] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

[7] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. In *Proc. of ICCV'07*, 2007.

[8] Adrian Barbu and Song-Chun Zhu. Generalizing Swendsen-Wang to Sampling Arbitrary Posterior Probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8), 2005.

[9] Leon Barrett and Aleksandr Simma. MCMC With Disconnected State Spaces, 2005.

[10] Marian Stewart Bartlett, Javier R. Movellan, and Terrence J. Sejnowski. Face recognition by independent component analysis. *IEEE Trans. on Neural Networks*, 13(6), 2002.

[11] J. Bigun and G. Granlund. Optimal orientation detection of linear symmetry. In *Proc. of ICCV'87*, 1987.

[12] Michael J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *Proc. of ICCV'93*, 1993.

[13] David M. Blei, Andrew Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[14] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2011.

[15] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61:211–231, 2005.

[16] Liangliang Cao and Li Fei-Fei. Spatially coherent latent topic model for concurrent object segmentation and classification. In *Proc. of ICCV'07*, 2007.

[17] F Caron, Manuel Davy, and A Doucet. Generalized Polya Urn for Time-varying Dirichlet Process Mixtures. In *Proc. of UAI'07*, 2007.

[18] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond, 2003.

[19] Yeonseung Chung and David B. Dunson. The local Dirichlet Process. *Annals of the Inst. of Stat. Math.*, 63(1):59–80, 2009.

[20] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. of CVPR'00*, 2000.

[21] T. F. Cootes and C. J. Taylor. A mixture model for representing shape variation. *Image and Vision Computing*, 17(8):567–573, 1999.

[22] Timothy F. Cootes, D. Cooper, Christopher J. Taylor, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61:38–59, 1995.

[23] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:681–685, 2001.

[24] D. Demirdjian, T. Ko, and T. Darrell. Constraining human body tracking. In *Proc. of ICCV'03*, 2003.

[25] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:932–946, 2002.

[26] Tom Drummond and Roberto Cipollar. Real-time tracking of complex structures with on-line camera calibration. *Image and Vision Computing*, 20:427–433, 2002.

[27] S. Duane, A.D. Kennedy, B.J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2), 1987.

[28] Daniel Eaton and Kevin Murphy. Bayesian Structure Learning using Dynamic Programming and MCMC. In *Proc. of UAI'07*, 2007.

[29] A. Efros and William Freeman. Image quilting for texture synthesis and transfer. In *Proc. of SIGGRAPH'01*, 2001.

[30] Alexei A. Efros, Alexender C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *Proc. of ICCV'03*, 2003.

[31] Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. An HDP-HMM for Systems with State Persistence, 2008.

[32] N. Friedman and D. Koller. Being Bayesian about network structure - A Bayesian approach to structure discovery in Bayesian networks. *Machine learning*, 50(1):95–125, 2003.

[33] M. F.Tappen. Utilizing variational optimization to learn markov random fields. In *Proc. of ICCV'07*, 2007.

[34] M. F.Tappen, C. Liu, E. H. Adelson, and William Freeman. Learning gaussian conditional random fields for low-level vision. In *Proc. of CVPR'07*, 2007.

[35] Jan Gasthaus, Frank Wood, D. Görur, and Y.W. Teh. Dependent Dirichlet Process Spike Sorting. In *Proc. of NIPS'09*, 2009.

[36] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14, 1992.

[37] Lena Gorelick, Moshe Blank, Eli Shechtman, Michale Irani, and Ronen Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[38] Peter J. Green. Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, 82(4), 1995.

[39] P.J. Green. Trans-dimensional Markov chain Monte Carlo. *Highly structured stochastic systems*, 27, 2003.

[40] J. E Griffin and M. F. J Steel. Order-Based Dependent Dirichlet Processes. *Journal of the American Statistical Association*, 101(473):179–194, March 2006.

[41] Matthias Grundmann, V Kwatra, Mei Han, and Irfan Essa. Efficient Hierarchical Graph-based Video Segmentation. In *Proc. of CVPR'10*, 2010.

[42] Yanlin Guo, Steve Hsu, Harpreet S. Sawhney, Rakesh Kumar, and Ying Shan. Robust object matching for persistent tracking with heterogeneous features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:824–839, 2007.

[43] Firas Hamze and N. de Freitas. Intracluster Moves for Constrained Discrete-Space MCMC. In *Proc. of UAI'10*, 2010.

[44] Tony X. Han, Huazhong Ning, and Thomas S. Huang. Efficient nonparametric belief propagation with application to articulated body tracking. In *Proc. of CVPR'06*, 2006.

[45] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Proc. of NIPS'03*, 2003.

[46] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proc. of SIGIR'99*, 1999.

[47] Bernard K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[48] Xiaolei Huang and Dimitris N. Metaxas. Metamorphs: Deformable shape and appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1444–1459, 2008.

[49] Serdar Ince and Janusz Konrad. Occlusion-aware optical flow estimation. *IEEE Transactions on Image Processing*, 17:1443–1451, 2008.

[50] M. R. Jerrum and A. J. Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18:1149–1178, 1989.

[51] Nebojsa Jojic and Brendan J. Frey. Learning flexible sprites in video layers. In *Proc. of CVPR'01*, 2001.

[52] Michael Kass, Andrew Witkin, and Demetri Terzopoulous. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1988.

[53] Seyoung Kim and Padhraic Smyth. Hierarchical dirichlet processes with random effects. In *Proc. of NIPS'06*, 2006.

[54] Nathan Kitchen and Andreas Kuehlmann. A Markov Chain Monte Carlo Sampler for Mixed Boolean/Integer Constraints. In *Computer Aided Verification*, pages 446–461, 2009.

[55] Achim Klenke. *Probability Theory: A Comprehensive Course.* Springer, 2007.

[56] Iasonas Kokkinos and Alan Yuille. Unsupervised learning of object deformation models. In *Proc. of ICCV'07*, 2007.

[57] M. Pawan Kumar, P.H.S. Torr, and A. Zisserman. Learning Layered Motion Segmentations of Video. In *Proc. of ICCV'05*, 2005.

[58] Xiangyang Lan and D.P. Huttenlocher. A unified spatio-temporal articulated model for tracking. In *Proc. of CVPR'04*, 2004.

[59] John M. Lee. *Introduction to Smooth Manifolds.* Springer, 2002.

[60] Julien Lefevre and Sylvain Baillet. Optical flow and advection on 2-riemannian manifolds: A common framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1081–1092, 2008.

[61] Ido Leichter, Michael Lindenbaum, and Ehud Rivlin. Bittracker - a bitmap tracker for visual tracking under very general conditions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1572–1588, 2008.

[62] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times.* American Mathematical Society, 2008.

[63] Fei-Fei Li and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *Proc. of CVPR'05*, 2005.

[64] A. Likas, N.P. Galatsanos, and I.E. Lagaris. A spatially constrained mixture model for image segmentation. *IEEE Trans. on Neural Networks*, 16(2):494–498, 2005.

[65] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of Imaging Understanding Workshop*, 1981.

[66] Steven N. MacEachern. Dependent Nonparametric Processes. In *Proceedings of the Section on Bayesian Statistical Science*, 1999.

[67] Ravikanth Malladi, James Sethian, and Baba Vermuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:158–175, 1995.

[68] Abdol-reza Mansouri, Dipti Prasad Mukherjee, and Scott T. Acton. Constraining active contour evolution via lie groups of transformation. *IEEE Transactions on Image Processing*, 13:853–863, 2004.

[69] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Proc. of ICCV'09*, 2009.

[70] Amar Mitiche and Patrick Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *International Journal of Computer Vision*, 19(1):29–55, 1996.

[71] Francesc Moreno-noguer, Alberto Sanfeliu, and Dimitris Samaras. Dependent multiple cue integration for robust tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:670–685, 2008.

[72] Peter Muller, Fernando Quintana, and Gary Rosner. A Method for Combining Inference across Related Nonparametric Bayesian Models. *J. R. Statist. Soc. B*, 66(3):735–749, August 2004.

[73] Radford M. Neal. Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[74] Andrew Ng, M. I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. of NIPS'01*, 2001.

[75] Nikos Paragios and Rachid Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:266–280, 2000.

[76] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. Bilateral filtering: Theory and applications. *Foundations and Trends in Computer Graphics and Vision*, 4(1), 2008.

[77] Jian Pei, Haixun Wang, Jian Liu, Ke Wang, Jianyong Wang, and Philip S. Yu. Discovering frequent closed partial orders from strings. *IEEE Trans. on Knowledge and Data Engineering*, 18(11), 2006.

[78] Vinayak Rao and Yee Whye Teh. Spatial Normalized Gamma Processes. In *Proc. of NIPS'09*, 2009.

[79] Carl Edward Rasmussen. The Infinite Gaussian Mixture Model. In *Proc. of NIPS'00*, 2000.

[80] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[81] Yogesh Rathi, Namrata Vaswani, Allen Tannenbaum, and Anthony Yezzi. Tracking deforming objects using particle filtering for geometric active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1470–1475, 2007.

[82] Lu Ren, David B. Dunson, and Lawrence Carin. The Dynamic Hierarchical Dirichlet Process. In *Proc. of ICML'08*, 2008.

[83] Stefan Roth and Michael J. Black. Fields of experts: A framework for learning image priors. In *Proc. of CVPR'05*, 2005.

[84] Stefan Roth and Michael J. Black. Steerable random fields. In *Proc. of ICCV'07*, 2007.

[85] Stefan Roth and Michael J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, 2008.

[86] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[87] K. G. Samuel and M. F.Tappen. Learning optimized map estimates in continuously-valued mrf models. In *Proc. of CVPR'09*, 2009.

[88] U. Schmidt, Q. Gao, and S. Roth. A generative perspective on mrfs in low-level vision. In *Proc. of CVPR'10*, 2010.

[89] Thomas Schoenemann and Daniel Cremers. High resolution Motion Layer Decomposition using Dual-space Graph Cuts. In *Proc. of CVPR'08*, 2008.

[90] J. Sethuraman. A Constructive Definition of Dirichlet Priors. *Statistica Sinica*, 4(2):639–650, 1994.

[91] Eli Shechtman and Michale Irani. Space-time behavior-based correlation or how to tell if two underlying motion fields are similar without computing them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:2045–2056, 2007.

[92] Hedvig Sidenbladh and Michael J. Black. Learning image statistics for bayesian tracking. In *Proc. of ICCV'01*, 2001.

[93] C. Stauffer. Adaptive background mixture models for real- time tracking. In *Proc. of CVPR'99*, 1999.

[94] Chris Stauffer and Eric Grimson. Adaptive background mixture models for real-time tracking. In *Proc. of CVPR'98*, 1998.

[95] Deqing Sun, Stefan Roth, J.P. Lewis, and Michael J. Black. Learning optical flow. In *Proc. of ECCV'08*, 2008.

[96] Deqing Sun, Erik B Sudderth, and Michael J Black. Layered Image Motion with Explicit Occlusions, Temporal Consistency, and Depth Ordering. In *Proc. of NIPS'10*, 2010.

[97] R.H. Swendsen and J. Wang. Nonuniversal critical dynamics in monte carlo simulation. *Physics Review Letters*, 58(2), 1987.

[98] M. Tanaka and M. Okutomi. Locally adaptive learning for translation-variant mrf image priors. In *Proc. of CVPR'08*, 2008.

[99] Yee Whye Teh. Dirichlet Process, 2007.

[100] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[101] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analysis. *Neural Computation*, 11(2), 1999.

[102] M. Tipping and C. Bishop. Probabilistic principal component analysis. *J. R. Statist. Soc. B*, 61, 1999.

[103] Zhuowen Tu and Song-Chun Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 2002.

[104] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[105] Matthew Turk and Alex Pentland. Face recognition using eigenfaces. In *Proc. of CVPR'91*, 1991.

[106] G Vahedi, I V Ivanov, and E R Dougherty. Inference of Boolean networks under constraint on bidirectional gene relationships. *IET systems biology*, 3(3):191–202, 2009.

[107] Rene Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (gpca). *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, 2005.

[108] B. Walsh. Markov Chain Monte Carlo and Gibbs Sampling, 2002.

[109] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transaction on Image Processing*, 3(5):625–638, 1994.

[110] John Y. Wang and Edward H. Adelson. Representing Moving Images with Layers. *IEEE Transactions on Image Processing*, 3(5):625–38, 1994.

[111] Wei Wei, Jordan Erenrich, and Bart Selman. Towards Efficient Sampling : Exploiting Random Walk Strategies. In *Proc. of AAAI'04*, volume 000, 2004.

[112] Joachim Weickert and Christoph Schnörr. A theoretical framework for convex regularizers in pde-based computation of image motion. *International Journal of Computer Vision*, 45:245–264, 2001.

[113] Yair Weiss. Smoothness in Layers: Motion Segmentation using Nonparametric Mixture Estimation. In *Proc. of CVPR'97*, 1997.

[114] Yair Weiss and Edward H. Adelson. A Unified Mixture Framework for Motion Segmentation: Incorporating Spatial Coherence and Estimating the Number of Models. In *Proc. of CVPR'06*, 2006.

[115] Yair Weiss and William Freeman. What makes a good model of natural images? In *Proc. of CVPR'07*, 2007.

[116] Greg Welch and Gary Bishop. An introduction to the kalman filter. In *SIGGRAPH 2001 Course*, 2001.

[117] Ming-Hsuan Yang. Kernel eigenfaces vs kernel fisherfaces: Face recognition using kernel methods. In *Prof. of International Conf. on Automatic Face and Gesture Recognition*, 2002.

[118] Yue Zhou and Hai Tao. A Background Layer Model for Object Tracking through Occlusion. In *Proc. of CVPR'03*, 2003.

[119] Songchun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory of texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

[120] Xiaojin Zhu and John Lafferty. Time-Sensitive Dirichlet Process Mixture Models, 2005.