GENERALIZED DISTRIBUTED LEARNING UNDER UNCERTAINTY FOR CAMERA NETWORKS

By

SEJONG YOON

A dissertation submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Vladimir Pavlovic

And approved by

New Brunswick, New Jersey

October, 2016

ABSTRACT OF THE DISSERTATION

Generalized Distributed Learning Under Uncertainty for Camera Networks

by SEJONG YOON

Dissertation Director:

Vladimir Pavlovic

Consensus-based distributed learning is a machine learning technique used to find the general consensus of local learning models to achieve a global objective. It is an important problem with increasing level of interest due to its applications in sensor networks. There are many benefits of distributed learning over traditional centralized learning, such as faster computation and reduced communication cost. In this dissertation, we focus on the merit that distributed learning can be performed in a fully decentralized way, which makes it one step further different from parallel computing approaches.

First, we propose a general distributed probabilistic learning framework based on distributed optimization using an Alternating Direction Method of Multipliers (ADMM). We show that it can be applied to computer vision algorithms which have traditionally assumed a centralized computational setting. We demonstrate that our probabilistic interpretation of the decentralized processing is useful in dealing with missing values which are not explicitly handled in prior works. We provide empirical evaluations on a computer vision problem termed distributed affine structure from motion (SfM).

Second, we propose two useful extensions of the distributed probabilistic learning framework. We first extend our framework so that it can incrementally update the learned model in an online fashion. To do this, we propose to use a Bayesian inference model based on Bregman ADMM (B-ADMM). Next, we show that the distributed learning tasks can be carried out more rapidly by introducing smart update strategies to the underlying ADMM optimization algorithm. By adaptively balancing primal and dual residuals of ADMM, we demonstrate an improved empirical convergence speed in a fully decentralized setting, without limiting the application range of ADMM-based optimization.

Finally, we introduce a potential application of consensus-based distributed optimization on the human trajectory estimation problem. We formulate the trajectory estimation problem as a global optimization issue with constraints encoding various prior conditions that can be either allowed or forbidden in real world situations. We show that our method can effectively estimate the noisy, corrupted trajectories from off-the-shelf human trackers that could assist in human crowd analysis and simulation.

Acknowledgements

First of all, I should mention that I am very fortunate to have Prof. Vladimir Pavlovic as my doctoral dissertation adviser. I am so much indebted to him in nearly every aspect of my graduate student life at Rutgers, including, but not limited to, admission, courses, research, teaching, and even job search. Without his guidance and help, this dissertation would have never been possible to be completed.

I would like to express my special thank you to Prof. Mubbasir Kapadia, a passionate, intellectual researcher, whose guidance helped me conduct the important last part of this dissertation. I also would like to thank my committee members, Prof. Dimitris Metaxas and Prof. Norman I. Badler for their detailed, and insightful comments that significantly improved this dissertation.

Over the six years of my life at Rutgers, I was able to survive through challenging times with helps from many people that I cannot list them all here. I would like to thank Prof. William Steiger, who has been the graduate program director for most of my graduate years, for his crucial help in the beginning, and at the very end of my study. It has been my great pleasure to take Prof. Casimir Kulikowski's courses to learn insightful history and philosophy of artificial intelligence. I was also fortunate to get chances to learn about teaching in U.S. higher education from Dr. Sesh Venugopal. I thank all friends in CS, CBIM, and SEQAM group. I thank and wish the best luck to all my group-mates: Jongpil, Saehoon, Hai, Cuong, and Behnam. I also thank CS members of RKGSA community: Daehan, Daeyoung, Changkyu, Jae Woo, and Jay.

I would like to thank professors in Korea for helping me receive my doctorate degree: Prof. Saejoon Kim, Prof. Jongho Nang, and Prof. Kyung-Whan Oh.

Last, but not least, I would like to deeply thank my parents, Kwang Hak Yun and Yuna K. Paik for their unlimited support that made this dissertation possible.

Previous Publications. This dissertation, in part, contains works of the author and collaborators as presented in previous conference and workshop publications [1, 2, 3, 4, 5]. The author is the main contributor of the first and the last contribution of this thesis [1, 5], in terms of both theoretical and experimental aspects of the papers. For the first half of the second contribution [3], the author originally thought of the employment of the Bregman ADMM [6] into the framework, implemented the derived closed form updates, and conducted some of the experiments. For the other half of the second contribution [4], the author co-invented two of the three proposed algorithms, conducted some of the experiments, and wrote the manuscript.

Cliparts. Some cliparts used in figures in this dissertation were obtained from `openclipart.org` and I would like to extend my appreciation to the designers who contributed and shared their artworks for free and released them to the public domain.

Table of Contents

List of Tables

List of Figures

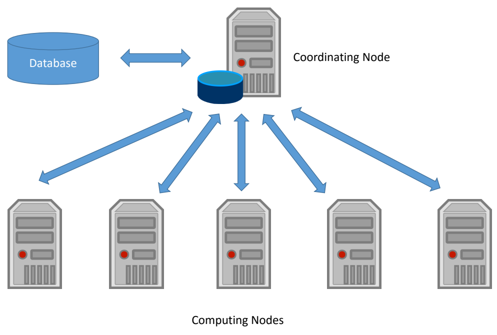
Chapter 1

Introduction

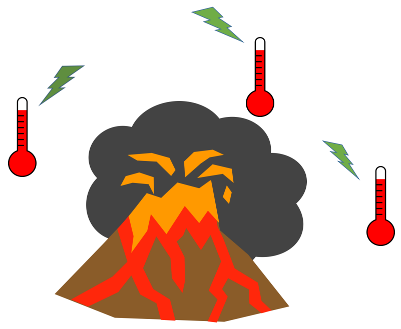
Advances in sensor technology have made many things that were once considered to be challenging or expensive to implement become feasible and affordable. For example, thousands of surveillance cameras are deployed to monitor different parts of large cities, hundreds of weather stations are collecting information nationwide, and many high-precision sensors are continuously monitoring extreme areas such as volcanos and ocean floors to predict potential natural disasters. The types of sensors are not necessarily limited to fixed devices. Millions of mobile devices are equipped with sensors such as high resolution cameras, gyroscopes, accelerometers, thermometers, and global positioning system (GPS) receivers, and are sold around the world everyday.

The aforementioned sensors are input sources of an enormous amount of data that needs to be processed and analyzed. This large-scale data processing is carried out at multiple levels. At a low level, this involves continuous recording of raw measurements from sensors. At a mid level, this involves light weight processing of the raw data, e.g., detecting or tracking objects from visual sensors. At a high level, data processing is done in scale, considering input from multiple sensors at the same time, in order to provide high level summarized information. These processing tasks, particularly at a high level, demand significant computational power due to the size of the data. Naturally, it is desirable to have scalable methods and algorithms to handle such large-scale data both efficiently and accurately. Note that mid-level and high-level processing also depend on low-level processing results, thus such methods are required to be robust to individual sensor failure or noise caused by environmental changes.

In this thesis, we consider some of the important problems in this domain, and show how *distributed machine learning* methods can be used to solve those problems.



(a) Deliberately / Induced distributed data



(b) Inherently / Intrinsically distributed data

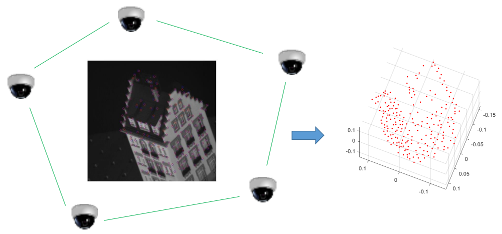
Figure 1.1: Example problem scenarios in consensus-based distributed learning based on how input data is distributed.

1.1 Motivation

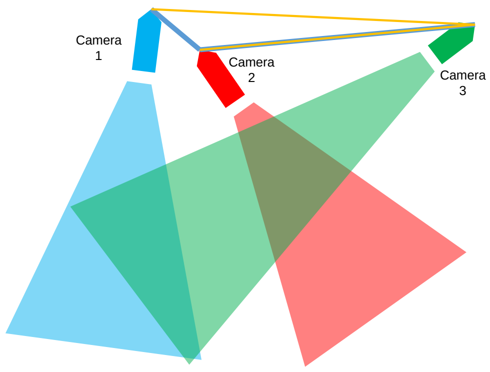
There are two different types of large-scale data that are distinguishable based on how it is obtained and processed. One type of large-scale data is *deliberately* or *induced* distributed data. This is a typical setting in so-called big data analysis problems where we have enormous amount of data that simply do not fit into the memory of a single machine. To do any meaningful analysis on this data, one has to divide it into a manageable size and distribute it into computing nodes, as shown in Figure 1.1a. In this case, there must be a central coordinating node that maps the data to local nodes and then aggregates the results back to the coordinating node to obtain the summary. The movie rating prediction challenge by Netflix [7] and other KDD cup challenges are good examples of this type of distributed data. While this problem of how to distribute data and aggregate the result back efficiently and robustly is an important topic, we are mainly interested in the other type of distributed data in this thesis.

The other type of the large-scale data is *inherently* or *intrinsically* distributed data. For example, suppose we deployed a large set of thermo sensors to monitor a volcano, as shown in Figure 1.1b. In this case, the measurements collected by each thermo sensor node are located far apart from other sensor nodes. Thus, the data itself is distributed over a wide area and hence collected in a distributed way. In this case, we identify several challenging issues due to this nature of data locality. One such challenge is the communication overhead. If all sensors try to directly connect to a central server and transmit their measurement information, it requires significant communication bandwidth and transmission time. The situtation becomes worse if the communication is to be done through wireless networks. If the monitoring area is an extreme environment, as in the aforementioned example (large volcano), the issue becomes more severe because (a) sensors need to establish a direct communication channel to the server located far away from them, (b) putting repeater towers in such extreme regions is not desirable and (c) using satelite communication may be expensive in terms of electric power consumption. The second challenge is the real-time or anytime processing [8] requirements for large area sensing systems [9]. As many sensor networks need real-time processing, it may be too late to make any decision only after sending all local information to the server and processing them. An example of such cases is the tracking problem. Assume that we have sensors with enough computing power to track objects. If one tracker starts tracking a target and lost it at some point, it is reasonable and probably most efficient to begin searching for the target using neighboring sensors. In such cases, algorithms that can work in a *decentralized* fashion would have merit over centralized processing.

In the case of visual sensor networks, i.e., camera networks, there are additional challenges in addition to those innate from the general sensor networks. Traditional computer vision algorithms, particularly those that exploit various probabilistic and learning-based approaches, are often formulated in centralized settings. A scene or an object is observed by a single camera with all acquired information centrally processed and stored in a single knowledge base (e.g., a classification model). Even if the problem setting relies on multiple cameras, as may be the case in multi-view or Structure from Motion (SfM) tasks [10] as shown in Figure 1.2a, all collected information is still processed and organized in a centralized fashion. However, modern computational settings are becoming increasingly characterized by networks of peer-to-peer connected devices, with local data processing abilities. Nevertheless, the overall goal of such distributed camera networks may still be to exchange information and form a consensus interpretation of the visual scene. For instance, even if a camera observes a limited set of object



(a) Multi-Camera Structure from Motion



(b) Vision-Graph mismatch [10]

Figure 1.2: Example problems in distributed learning for camera networks.

views, one would like its local computational model to reflect a general 3D appearance of the object visible by other cameras in the network. Thus, a distributed learning approach, where the cameras try to achieve *a common goal* shared across all cameras, is desirable in such scenarios.

It is worth noting that in many cases, the cameras may have overlapped views that are inconsistent with their physical locations as shown in Figure 1.2b. In the figure, camera 1 and 2 are physically closely located neighbors, thus they should be neighbors when communicating. In this way, they can reduce the power consumption for information transmission during any learning process. However, what camera 1 is monitoring in fact overlaps with the field of view of camera 3. Thus, there is a discrepency in the physical connectivity (blue lines) and the visual correlation (orange lines). This is called vision-graph mismatch [10] and it is the one of the reasons why distributed learning for computer vision problems are challenging. Moreover, unlike simpler sensors such as thermometers, it is not trivial to find correspondence between image frames observed from different cameras. There are other factors, such as illumination change or occulusion, that can fail even state-of-the-art algorithms associated with this problem.

Therefore, it is clearly both desirable and challenging to design distributed learning methods for large-scale sensor networks, either visual or non-visual. In this thesis, we formulate this distributed learning as a *consensus-based* distributed optimization problem to deal with the challenges. We consider two aspects of *learning* in this thesis

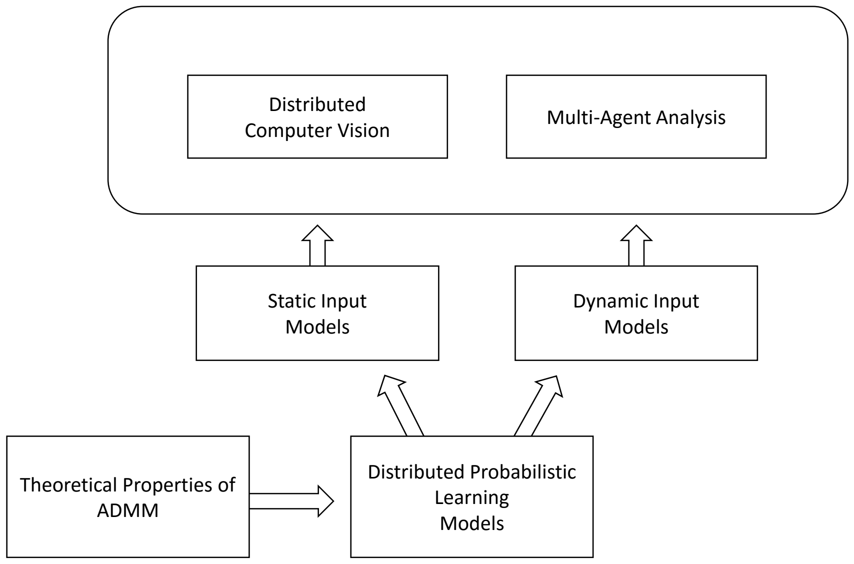

Distributed Computer Vision
Multi-Agent Analysis
Static Input Models
Dynamic Input Models
Theoretical Properties of ADMM
Distributed Probabilistic Learning Models

Figure 1.3: Overview of the dissertation organization

in consensus-based distributed optimization. One is the distributed *state* estimation, in which the nodes try to estimate the states of the sensors or targets based on local observations, while the other is the distributed *model* estimation, in which the nodes share a common parametric model as their belief to reach a consensus. Many examples described earlier, such as the problem of estimating the mean temperature of distributed thermometers, belong to the category of state estimation. We will see how a parametric probablistic model can be estimated in a decentralized way for distributed data obtained by camera networks in the following chapters.

1.2 Dissertation Statement and Contributions

To address the aforementioned problems, we propose a consensus-based, generalized distributed probabilistic learning framework as a solution. We first define the consensus-based distributed learning problem we consider in this thesis, then provide a literature review on related works in Chapter 2. Then we introduce our distributed learning framework and its extensions and appliations in subsequent chapters. Figure 1.3 depicts the overview of the dissertation contributions. Specifically, we will be following three main contributions in this dissertation:

- In Chapter 3, we propose a general probabilistic learning framework that can learn a probabilistic model, which works in a fully decentralized fashion. As an example application of the framework, we derive a distributed counterpart of the probabilistic principal component analysis (PPCA) [11]. We show how the distributed PPCA algorithm can effectively be applied to a distributed computer vision problem, the distributed affine SfM. We demonstrate that the proposed distributed algorithm can robustly reconstruct the shared 3D structure in camera networks even under the presence of noise and missing information, both in the case of missing at random and missing not at random cases.

- In Chapter 4, we propose two extensions of our framework. First, we extend our framework to an online algorithm so that it can handle dynamically changing input data by reformulating it into a distributed Bayesian learning model. With the Bayesian formulation, we can naturally update the model in an online fashion and additionally, obtain richer information from the parameters by inferring on the full posterior distribution rather than the non-Bayesian formulations that have been typical choices in prior works in distributed learning. Second, we propose extensions to the underlying distributed optimization algorithm so that it can empirically converge more quickly. We demonstrate that the proposed extensions can speed up the empirical convergence of the distributed SfM problem.

- In Chapter 5, we propose a novel, potential application of consensus-based learning for a crowd analysis problem. Motivated by prior works in multi-agent system literature [12, 13], we show how consensus-based optimization problem formulation can be used to estimate optimal, desired crowd trajectories with prior constraints encoded into a global optimization-based framework. Using simulated crowd trajectories generated by standard scenarios in crowd analysis literature, we demonstrate how the proposed consensus-based optimization method can be used in crowd trajectory analysis problems.

Finally, we draw conclusions and discuss promising future research directions and open problems in Chapter 6.

Chapter 2

Distributed Learning: Problem Description and Review

In this chapter, we first define what we mean by *distributed learning* in this thesis, and then introduce related prior works in the literature, focusing on distributed optimization methods and applications in sensor networks.

2.1 Problem Description

We formally define the type of distributed learning problems we consider in this thesis. In general, we formulate the problems as a *consensus-based* optimization problem [14]. A general consensus-based optimization problem can be written as:

$$\begin{aligned} \arg\min_{x_i} \quad & \sum_{i=1}^{J} f_i(x_i) \\ s.t. \quad & x_i = x_j, \quad \forall i \neq j, \end{aligned} \tag{2.1}$$

that is we want to find the set of optimal parameters x_i, $i, j = 1..J$ that minimizes the sum of convex objective functions $f_i(x_i)$, where J denotes the total number of functions. This problem is typically a reformulation of a centralized optimization task:

$$\arg\min_{x} \ f(x) \tag{2.2}$$

with a decomposable objective:

$$f(x) = \sum_{i=1}^{J} f_i(x). \tag{2.3}$$

Given the consensus formulation, the original problem can be solved by decomposing the problem into J subproblems, so that J processors can cooperate to solve the overall problem by changing the equality constraint to $x_i = \bar{x}$, where $\bar{x}$ denotes a globally shared parameter.

2.2 Distributed Optimization and Learning

In this section, we review the underlying optimization methods that enable distributed learning in general, including the consensus-based optimization problem of concern.

2.2.1 Parallel vs. Distributed Learning

In distributed machine learning and optimization literature, the terms *parallel* and *distributed* are sometimes used to imply different meanings. Therefore, it is important to clarify them at this stage. In some prior works, *parallel* computation means that multiple processing nodes reside in a single machine, while *distributed* computation implies there are multiple machines that must communicate over the network. Both of these settings require a central coordinating node to map the data to multiple workers and aggregate the results back, thus they belong to the *induced distributed* case based on our categorization of the problem. On the other hand, there are prior works aimed for *intrinsically distributed* case, and they also use the term *distributed* to refer to their decentralized methods. Since our focus is on distinguishing between induced and intrinsically distributed cases, we refer to the former as *parallel* and the latter as *distributed* throughout this thesis.

This classification of terms is consistent with how they are defined in representative literature on this topic [14], which defines *parallel* systems as “systems consist of several processors that are located within a small distance of each other” and *distributed* systems as “(systems with) processors may be far apart, and interprocessor communication is more problematic and communication delays may be unpredictable and the communication links may be unreliable.” Most importantly, “they are usually loosely coupled; there is very little, if any, central coordination and control.” We will use the term *decentralized* when we need to emphasize the computation of a *distributed* setting can be done in a fully decoupled way.

A significant number of prior works in machine learning and optimization literature discuss the parallel case involving efforts being made to scale up the amount of data that can be processed by traditional centralized machine learning methods. Nevertheless, we briefly review those works in the parallel setting, as they will provide precursors for our distributed framework that functions in a fully decentralized way. In fact, several algorithmic issues of the two settings are similar and closely related [14].

2.2.2 Optimization for Parallel and Distributed Learning

In optimization literature, finding an optimal solution of a given objective function with multiple workers has been an active topic for many decades [15]. The parallelization of an optimization problem can be performed either by the optimization method structure or by a transformation of the problem structure [14]. To illustrate the former, assume that we want to minimize

$$\begin{aligned} &\underset{\mathbf{x}=(\mathbf{x}_1,\mathbf{x}_2,\cdots\mathbf{x}_J)}{\arg\min} \quad f(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_J) \\ &\qquad s.t. \quad \mathbf{x}\in\Omega=\prod_{i=1}^{J}\Omega_i, \quad \Omega\subseteq\mathbb{R}^M, \quad \mathbf{x}_i\in\mathbb{R}^{M_i}, \quad \sum_{i=1}^{J}M_i=M \end{aligned} \tag{2.4}$$

where $f:\mathbb{R}^M\to\mathbb{R}$ is a continuously differentiable cost function and Ω_i are closed convex sets. This is a prototypical example of a (centralized) state estimation problem. On the other hand, if the variables $\mathbf{x}_i$ depend on an unknown shared parameter θ that needs to be found, then the problem becomes a model estimation problem.

For this type of problem, we can apply gradient descent algorithms (Jacobi, Gauss-Seidel or (approximate) Newton method if f is twice differentiable) to find the $\mathbf{x}$ minimizing f. Or, we can also use so-called nonlinear block-coordinate descent methods. The basic idea of these techniques is to iteratively find one variable at a time, fixing the others. In the case of nonlinear Jacobi, each variable is updated as

$$\mathbf{x}_i^{(t+1)}=\underset{\mathbf{x}_i}{\arg\min}\; f(\mathbf{x}_1^{(t)},\cdots,\mathbf{x}_{i-1}^{(t)},\mathbf{x}_i,\mathbf{x}_{i+1}^{(t)},\cdots,\mathbf{x}_J^{(t)}), \tag{2.5}$$

while in the case of nonlinear Gauss-Seidel, each variable is updated as

$$\mathbf{x}_i^{(t+1)}=\underset{\mathbf{x}_i}{\arg\min}\; f(\mathbf{x}_1^{(t+1)},\cdots,\mathbf{x}_{i-1}^{(t+1)},\mathbf{x}_i,\mathbf{x}_{i+1}^{(t)},\cdots,\mathbf{x}_J^{(t)}). \tag{2.6}$$

Thus, nonlinear Jacobi requires information of all other variables' past states, while nonlinear Gauss-Seidel requires the update to be carried out sequentially. The order of variable updates in the nonlinear Gauss-Seidel can be different in each iteration. All these methods are guaranteed to converge to the minimum of f as long as f is continuously differentiable and convex [14]. They are also methodologically well suited for parallel computation, since the variable updates can be done simultaneously with a carefully designed synchronization scheme. Therefore, as long as the problem is in the form of (2.4), one can obtain a parallel computation algorithm to solve it efficiently.

However, this is not always the case. For example, if the constraint set is coupled so that it is not in the form of the Cartesian product of simpler sets, the aforementioned algorithms are not directly applicable. In this case, we need to transform the problem structure so that it can be solved in a parallel (or distributed) way. One line of such approaches are the *decomposition* methods. These methods, based on the duality theory (details on this theory can be found in [16] or Appendix C of [14]), have been in use since the 1960s, including Dantzig-Wolfe [17] and Benders decompositions [18]. The basic idea of these methods is to separate the original problem into several simpler subproblems, and then coordinate them with a master problem.

The problems to be decomposed need to be in a specific form, and there are two types of decomposition methods: primal and dual. The former methods are appropriate when the variables are coupled, with the latter being useful when the constraints are coupled, although this is not a strict rule [19]. We give an example problem of the dual decomposition methods here. Comprehensive tutorials on decomposition methods can be found in [19, 20]. Assume that we are given the following constrained optimization problem:

$$\arg\min_{\mathbf{x}} \quad \sum_{i=1}^{J} f_i(\mathbf{x}_i), \quad s.t. \quad \sum_{i=1}^{J} \mathbf{A}_i \mathbf{x}_i = \mathbf{b}, \tag{2.7}$$

where $f_i : \mathbb{R}^D \to \mathbb{R}$, $\mathbf{A}_i \in \mathbb{R}^{D\times M}$ and $\mathbf{b} \in \mathbb{R}^D$. One can see that this problem would be fully decomposable if it were an unconstrained problem. Thus, we use the Lagrangian relaxation on (2.7) as

$$\arg\min_{\mathbf{x}} \quad \sum_{i=1}^{J} f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^\top \left(\sum_{i=1}^{J} \mathbf{A}_i \mathbf{x}_i - \mathbf{b} \right), \tag{2.8}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^D$ is the dual variable. Now the problem is separable, as the subproblem

$$\arg\min_{\mathbf{x}_i} \quad f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^\top \mathbf{A}_i \mathbf{x}_i, \tag{2.9}$$

can be solved easily, and we solve the dual problem

$$\arg\max_{\boldsymbol{\lambda}} \quad g(\boldsymbol{\lambda}) = \sum_{i=1}^{J} g_i(\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \mathbf{b}, \tag{2.10}$$

as the master problem, where $g_i(\boldsymbol{\lambda})$ is the dual function obtained by minimizing (2.9) with given $\boldsymbol{\lambda}$. The master problem is typically solved by using subgradient methods [20].

Another line of approach transforming the problem structure are the augmented Lagrangian methods, which also use the duality theory. A benefit of the augmented Lagrangian methods over simple decomposition techniques is that they are guaranteed to find a primal solution even when f is not strictly convex [21]. In the previous example, if functions f_i were strictly convex, then there would be a unique primal solution for (2.9) for given $\boldsymbol{\lambda}$. Thus, after finding the solution for the master problem (2.10), we can find the optimal primal solution in subproblems in this case. In other words, strict convexity of the objective function implies the differentiability of the dual function [14]. There are ways to solve the problem when the dual is not differentiable [22], but the augmented Lagrangian methods relax the assumption of strictly convex into a milder one, i.e., convex.

We begin our introduction of the augmented Lagrangian methods with the method of multipliers [23, 24]. Consider the constrained optimization problem

$$\arg\min_{\mathbf{x}} \quad f(\mathbf{x}), \quad s.t. \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.11}$$

where $f : \mathbb{R}^D \to \mathbb{R}$ is a closed convex function, $\mathbf{A} \in \mathbb{R}^{D \times M}$, $\mathbf{x} \in \mathbb{R}^M$ and $\mathbf{b} \in \mathbb{R}^D$. Using the Lagrangian relaxation and an additional squared penalty term, we can obtain an unconstrained objective, called the augmented Lagrangian function

$$\mathcal{L}_\eta(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{\eta}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2, \tag{2.12}$$

where $\eta > 0$ is called the parameter of the augmented Lagrangian. The method of multiplers is the sequential minimization of the two iterative updates

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x}} \mathcal{L}_{\eta^t}(\mathbf{x}, \boldsymbol{\lambda}^{(t)}), \tag{2.13}$$

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \eta \left(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}\right), \tag{2.14}$$

One can show that the method of multipliers is actually the proximal minimization algorithm [25, 14]. Since the squared penalty ensures strict convexity, we obtain the aforementioned benefit.

However, one cannot directly parallelize the squared penalty term even if f is separable, thus the method of multipliers is not the best option for developing a parallel or distributed optimization algorithm. Therefore, several modifications to the method were proposed, with one of them being the alternating direction method of multipliers (ADMM) [26, 27]. The algorithm solves problems in the form

$$\min \quad f(\mathbf{x}) + g(\mathbf{y}), \quad s.t. \quad \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{c}, \tag{2.15}$$

where $f : \mathbb{R}^{D_x} \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^{D_y} \to \mathbb{R} \cup \{+\infty\}$ are closed, proper convex functions, $\mathbf{x} \in \mathbb{R}^{D_x}$ and $\mathbf{y} \in \mathbb{R}^{D_y}$ are variables and $\mathbf{A} \in \mathbb{R}^{D_c \times D_x}$, $\mathbf{B} \in \mathbb{R}^{D_c \times D_y}$ and $\mathbf{c} \in \mathbb{R}^{D_c}$ are known, with D_x, D_y, D_c denoting the dimension of the corresponding variables. ADMM solves the problem by iterative updates

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x}} \mathcal{L}_{\eta}(\mathbf{x}, \mathbf{y}^{(t)}, \boldsymbol{\lambda}^{(t)}), \tag{2.16}$$

$$\mathbf{y}^{(t+1)} = \arg\min_{\mathbf{y}} \mathcal{L}_{\eta}(\mathbf{x}^{(t+1)}, \mathbf{y}, \boldsymbol{\lambda}^{(t)}), \tag{2.17}$$

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \eta \left(\mathbf{A}\mathbf{x}^{(t+1)} + \mathbf{B}\mathbf{y}^{(t+1)} - \mathbf{c}\right), \tag{2.18}$$

where

$$\mathcal{L}_{\eta}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\lambda}^{\top} (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}) + \frac{\eta}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}\|_2^2 \tag{2.19}$$

is the augmented Lagrangian. Note that the algorithm is similar to the Gauss-Seidel block-coordinate descent method we discussed earlier. Theoretical results as well as a survey on applications of ADMM can be found in [14, 15] and references therein. A brief introduction is also provided in Appendix A of this thesis.

The ADMM algorithm is interesting, as it is highly parallelizable when we apply it to the separable problems such as (2.1). In order to obtain the parallel algorithm to solve the consensus problem (2.1), we introduce a globally shared variable z and reformulate the problem as an equivalent problem [15], as

$$\arg\min_{x_i, z} \quad \sum_{i=1}^{J} f_i(x_i), \quad s.t. \quad x_i - z = 0, \quad \forall i. \tag{2.20}$$

This problem can be solved by a series of ADMM updates

$$x_i^{(t+1)} = \arg\min_{x_i} \left\{ f_i(x_i) + \lambda_j^{(t)} \left(x_i - z^{(t)} \right) + \frac{\eta}{2} \left(x_i - z^{(t)} \right)^2 \right\}, \tag{2.21}$$

$$z^{(t+1)} = \frac{1}{J} \sum_{i=1}^{J} \left\{ x_i^{(t+1)} + \left(\frac{1}{\eta} \right) \lambda_i^{(t)} \right\}, \tag{2.22}$$

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \eta \left(x_i^{(t+1)} - z^{(t+1)} \right). \tag{2.23}$$

Of course, this algorithm is not a distributed algorithm, since it requires a coordinating center. It is possible, however, to devise a fully decentralized distributed optimization algorithm based on the ADMM [28, 21]. Actually, at least two ways of ADMM-based algorithms have been proposed to devise general distributed learning algorithms. A good comparative analysis of the two methods, in an average consensus problem [29], can be found in [28]. Both methods introduce auxiliary variables to ease the derivation. In the first method [30, 31], the consensus optimization problem (2.1) is formulated as

$$\arg\min_{x_i, z_j} \quad \sum_{i=1}^{J} f_i(x_i), \quad s.t. \quad x_i = z_j, \quad \forall i \in \mathcal{V}, \quad j \in \mathcal{B}_i \tag{2.24}$$

where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the network with the set of vertices $\mathcal{V}$, the set of edges $\mathcal{E} = \{(i, j)\}$, and $\mathcal{B}_i$ denotes the set of one-hop neighbors of node i. In the second method [32, 33], the same consensus problem is formulated as

$$\arg\min_{x_i, z_{ij}} \quad \sum_{i=1}^{J} f_i(x_i), \quad s.t. \quad x_i = z_{ij}, \quad z_{ij} = x_j, \quad \forall i \in \mathcal{V}, \quad j \in \mathcal{B}_i. \tag{2.25}$$

The key difference is that (2.24) has two variables per edge (i, j) while (2.25) has only one. This makes the number of communication steps between node pairs per iteration to be two for the former, and one for the latter [34]. While the numbers of iterations needed for convergence are nearly equal [28], the latter algorithm is also more resilient to noise than the former [28]. The formulation has also been successfully applied to distributed classification and clustering [35, 36]. To this end, we employee the second method to devise the general distributed probabilistic learning framework in this thesis.

There have been other distributed optimization algorithms for consensus-based optimization problems, but we omit them here in order to focus on devising the general distributed probabilistic learning framework with applications in visual sensor networks. For readers interested in recent advances in the distributed optimization algorithms, [21] provides a survey and a good list of references, along with an original proposal of communication efficient distributed optimization algorithms based on ADMM, with several applications including compressed sensing.

2.3 Distributed Learning Applications

The consensus-based distributed learning problem can be found in various forms in different areas. For example, in robotics, the consensus-based distributed learning problem is termed "consensus problems in multi-agent coordination" [37]. In this section, we consider related prior works on consensus-based distributed learning applications found in machine learning and sensor network literature, explained as a high-level overview. Particularly, we focus on those in visual sensor networks, i.e., camera networks.

2.3.1 Distributed Learning in Machine Learning

In so-called "big data" related machine learning literature, significant efforts have been made in the context of data mining and knowledge discovery for large-scale databases [38, 39, 40]. These earlier works, by nature, are for the induced distributed case, and thus a parallel setting. The trend is similar in a recent comprehensive collection for large-scale machine learning [41] where the authors used the terms *parallel* and *distributed* to denote the difference in implementation aspect of two parallel settings, i.e., the distinction is based on whether the algorithm used OpenMP (processors reside in one machine; multi-threaded applications) or Message Passing Interface (processors reside in multiple machines and need data communication between them). While advances in large-scale data processing frameworks, including the popular MapReduce [42] algorithm are important, these methods are designed for parallel processing of induced distributed data, so we refer interested readers to the recent collection [41] for further references.

In the context of statistical, probabilistic machine learning, parallel approximate inference methods for latent variable models have gained interest in the past two decades, mainly due to the increased availability of large-scale text data (via web mining) and affordable computational power to process the data. Not surprisingly, parallel topic models [43] have become very popular, and it is easy to find recent studies in this area, e.g., [44, 45, 46]. However, these approaches are mostly parallel models, as the approximation algorithm is based on sampling, which is not trivial to perform in a decentralized way.

On the other hand, distributed algorithms have been studied extensively in the signal processing and sensor network community. Applications can be found in localization [47], classification and clustering [36] for wireless sensor networks and cognitive radio [48], or more generally as adaptive networks [49]. The applications for cognitive radio are particularly worth noting, as the distributed cooperative sensing [50, 51] closely relates to key aspects of distributed learning, although the problem is not necessarily consensus-based.

In terms of probabilistic learning, models such as Gaussian mixture [36] and Kalman filter [52] for the (average) consensus problem have been proposed in the sensor network community over the past decade, due to the popularity of those models in the areas' applications. Still, with a few exceptions that explicitly deal with the distributed setting (typically in wireless sensor networks, e.g., [9]), many works require one or more fusion centers that coordinate the processing. Moreover, if we further investigate Bayesian treatment of the learning models, there are few works that can perform the distributed learning in a fully decentralized way, since many of those Bayesian models depend on sampling methods. In chapter 4, we will see how the other approximation method, variational inference, can be executed in our decentralized probabilistic framework.

Lastly, it is worth mentioning the relationship of distributed learning to deep neural networks. Such networks require a large computational power, in order to find millions of parameters. Thus, it is inevitable for researchers to delve into the development of optimization algorithms that can scale. While the major contributions so far have focused on parallelization of neural networks by partitioning the network [53], it would be interesting to see how deep learning can be performed in a decentralized way.

2.3.2 Distributed Learning for Visual Sensor Networks

A number of distributed algorithms have been proposed to address the distributed learning problems in visual sensor networks, such as calibration, pose estimation, tracking, and object and activity recognition in large camera networks [10, 54, 55, 56, 57]. In order to deal with high dimensionality of vision problems, distributed latent space search methods, such as decentralized variants of PCA, have been studied in [58, 59]. A more general framework using distributed least squares [60] based on distributed averaging of sensor fusions [61] was introduced for PCA, triangulation, pose estimation and SfM. Similar approaches have been extended to settings such as the distributed object tracking and activity interpretation [52, 62, 63]. Even though the methods such as PCA or Kalman filtering have their well known probabilistic counterparts, the aforementioned approaches do not use probabilistic formulation when dealing with the distributed setting.

One critical challenge in distributed data analysis includes dealing with missing data. In camera networks, different nodes will only have access to a partial set of data features because of varying camera views or object movement. For instance, object points used for SfM may be visible only in some cameras and only in particular object poses. As a consequence, different nodes will be frequently exposed to missing data. However, most current distributed data analysis methods are algebraic in nature, and cannot seamlessly handle such missing data.

Often, prior works considered the distributed learning problems for camera networks in a non-Bayesian (often deterministic) fashion. Thus, the data is often assumed to be complete, i.e., not missing in individual nodes nor across the network. Therefore, such methods usually obtain point estimates of parameters by minimizing some loss function based on the complete data. However, Bayesian models have benefits of providing richer information on uncertainty of estimates and are robust to overfitting. Since these properties are desirable for machine learning methods for sensor networks that are susceptible to noise and sensor fail, it would be good to have our distributed learning model to be able to handle data in a Bayesian way.

2.3.3 Benchmark Datasets for Camera Networks

In this section, we briefly introduce what benchmark datasets are available in the context of distributed learning for camera networks. Since the same dataset can be used for different problems, we first briefly explain what are the goals and challenges in each of the problems using the cateogorization of [10], before introducing the datasets. We identify the following seven problems to fall into the category of distributed computer vision problems. We provide a short description for each of the problems, which are

- **Vision-Graph Discovery.** In Figure 1.2b, we showed a motivating example problem in distributed vision. To solve this problem, one has to establish correspondence between features extracted from each camera view.

- **Structure from Motion.** As introduced in the previous chapter, this problem involves the reconstruction of the object structure and camera pose, simultaneously using image measurements from multiple views. We focus on this problem in this thesis.

- **Camera Localization.** This problem is to estimate an individual camera's relative pose with respect to other cameras, based on image measurements obtained from multiple cameras in the network.

- **Calibration.** The above problems assume the cameras' intrinsic parameters (calibration matrix) are known. The problem of calibration is to estimate this parameter from image measurements. If the cameras are all of the same type, one may utilize the consensus-based formulation to obtain better estimates of the parameter.

- **Object Pose Estimation.** Given that cameras in the network are calibrated and their poses are known, the goal here is to cooperatively estimate the pose of 3D objects in the scene using image measurements from multiple cameras.

- **Activity Recognition.** Assuming the target is an intelligent moving object (e.g., person), this problem tries to find the correct activity label of the target's behavior. This is not a trivial extension of a single view-based activity recognition problem, because different views may yield varying activity labels. Thus, one has to resolve the potential discrepencies of the local algorithm output.

- **Object Tracking.** This is arguably one of the most active research problems in the camera network literature. The goal is to track one or more targets, in real-time, if possible, spanning multiple cameras' fields of views.

In the following, we introduce a few standard datasets. Some of them were designed for camera network (i.e., multiple cameras) problems, while others were not originally intended for the multiple-camera setting, but can be used as a simulated network.

Multi-View datasets. Some computer vision datasets are not originally designed for multiple cameras or camera-network problems, but they could be used as a simulated camera network for proof-of-concept experiments. For example, multi-view camera calibration, object recognition or pose estimation datasets such as Caltech 3D Objects on Turntable dataset [64] or EPFL Multi-View Car Dataset [65] are such examples. Both of these datasets have image frames of an object rotating on a turntable more or less of 360 degrees. One may partition the frames sequentially, i.e., the first 50 frames to be camera 1, the next 50 frames to be camera 2, etc. to simulate multiple cameras surrounding the target object in a circle. The EPFL Multi-View Car Dataset contains 20 different cars with image frames taken every approximately 3-4 degrees of rotation. The photos were taken by a Nikon D70 on a tripod at a motor show. The Caltech 3D Objects on Turntable dataset has several objects in the dataset, but only a handful of objects have image frames taken every degree. We use the Caltech dataset in our experiments in the following chapters[1].

[1]Unfortunately, this dataset is no longer available for download at the time of writing this thesis.

Hopkins115 [66] This dataset was not originally intended for the camera network setting, as it was designed for the motion segmentation problem, where the goal is to segment feature points extracted from video frames. However, as in the case of multi-view datasets, this dataset can be used for a simulated camera network configuration. It has been used for benchmark datasets for the problem of distributed affine structure from motion [60, 10]. Since we follow the same experimental setting as [60] in this thesis, we will use this dataset for the experiments.

Indoor Multi-Camera datasets. Some datasets use real multiple cameras, but the purpose of the dataset was not indended for "camera networks." For example, motion capture datasets, e.g., HumanEva [67] or multimodal datasets, e.g., 3DLife [68] use multiple cameras with overlapping fields of view. Often, cameras are synchronized and by the nature of the original purpose of the datasets, the datasets are relatively clean, i.e., have minimum level of noise. Of course, camera configurations in these datasets are rather unrealistic compared to the real world camera network, so the experiments using these datasets should be considered as simulated camera networks. Including a recently introduced dataset, MuHAVi [69], that provides 8 camera views, datasets in this category can be used for activity recognition problems.

PETS09 [70] This dataset is designed for the multiple pedestrian tracking problem using real multiple cameras. It is a dataset prepared for surveillance applications that had long existed as a popular benchmark dataset for the multi-target object tracking problem for a real multi-camera setting. The dataset contains recordings of 8 cameras taken at different times of the day, with different density and speed of controlled pedestrians. The dataset also provides camera calibration information.

EPFL [56, 71] This is another dataset for the multiple pedestrian tracking problem, using real multiple cameras. It contains video sequences captured by 3-4 cameras in 5 different environments. The pedestrians are controlled, with the exception of a basketball sequence, filming people playing a basketball game. The groundtruth as well as the calibration information are provided. Images were captured at 25 frames per second.

CamNeT [72] This dataset was designed for the pedestrian tracking problem under non-overlapping camera networks. It has six scenarios recorded at an university campus, using 5-8 cameras covering indoor and outdoor scenes. Sequences have varying conditions, such as illumination changes, complex topographies, and pedestrian densities and dynamics.

VideoWeb Activity [73] The VideoWeb activity dataset is taken from a long recording (2.5 hours) of multiple cameras containing dozens of activities with annotations. The sequences have different numbers of views of the activity.

RAiD [74] This dataset is for the pedestrian reidentification problem. 4 cameras (2 indoor and 2 outdoor) were used and 43 pedestrians were captured in the images.

3DPeS [75, 76, 77] This dataset is a general purpose multi-camera dataset mainly for human reidentification, but could be used for pedestrian detection, tracking, action recognition and trajectory analysis. Unlike other reidentification datasets, this dataset provides full video frames recorded over several days using 8 surveilence cameras. Pedestrians are notified about the cameras, but not controlled or instructed. However, the cameras are not synchronized and not all 8 cameras are used in the six sequences provided in the dataset, and only 1 sequence uses 3 cameras while others use only 2.

UvA Human Pose Estimation [78, 79] This dataset is for the object (human) pose estimation problem for overlapping camera networks. There are 3 synchronized cameras recording at 20 frames per second at VGA resolution. In total, 12 sequences were recorded and 17 body locations were annotated with 3D markers by human labeling. The labeling was done for one subject per sequence.

UvA Multi-Person Tracking [80] This dataset is also designed for multiple pedestrian tracking and activity / scene recognition problems, using image frames from overlapping cameras. There are 3 synchronized cameras recording 2 different environments at 20 frames per second. Controlled pedestrians and actors enact various activities. Calibration and ground truth annotations for tracking are provided.

Chapter 3

Generalized Distributed Probabilistic Learning

In this chapter, we propose a distributed consensus learning approach for parametric probabilistic models with latent variables that can effectively deal with missing data. This is the first example of a distributed model estimation problem we consider.

We assume that each node in a network can observe only a fraction of the data (e.g., object views in camera networks). Furthermore, we assume that some of the data features may be missing across different nodes. The goal of the network of sensors is to learn a single consensus probabilistic model (e.g., 3D object structure) without ever resorting to a centralized data pooling and centralized computation. We will demonstrate that this task can be accomplished in a principled manner by local probabilistic models and in-network information sharing, implemented as recursive distributed probabilistic learning.

In particular, we focus on probabilistic PCA (PPCA) as a prototypical example and derive its distributed version, the D-PPCA. We then suggest how missing data can be handled in this setting using a missing-data PPCA, and apply this model to solve the distributed SfM task in a camera network. Our model is inspired by the consensus-based distributed Expectation-Maximization (EM) algorithm for Gaussian mixtures [36], and we extend to deal with generalized linear Gaussian models [81]. Our model does not depend on any specific type of graphs. Our network, of arbitrary topology, is assumed to be static with a single connected component. These assumptions are reasonably applicable to many real-world camera network settings.

In Section 3.1, we first explain the general distributed probabilistic model. Section 3.2 shows how D-PPCA can be formulated as a special case of the probabilistic framework, and proposes the means for handling missing data. We then explain how

D-PPCA can be modified for the application in affine SfM. In Section 3.3, we report experimental results of our model using both synthetic and real data. Finally, we discuss our approach including its limitations and possible solutions in Section 3.4.

3.1 Distributed Probabilistic Learning Model

We start our discussion by first considering a general parametric probabilistic model in a centralized setting, and then we show how to derive its distributed form.

3.1.1 Centralized Setting

Let $\mathbf{X} = \{\mathbf{x}_n | \mathbf{x}_n \in \mathcal{R}^D\}$ be a set of i.i.d. multivariate data points with the corresponding latent variables $\mathbf{Z} = \{\mathbf{z}_n | \mathbf{z}_n \in \mathcal{R}^M\}$, $n = \{1, 2, ..., N\}$. Our model is a joint density defined on $(\mathbf{x}_n, \mathbf{z}_n)$ with a global parameter θ

$$(\mathbf{x}_n, \mathbf{z}_n) \sim p(\mathbf{x}_n, \mathbf{z}_n | \theta),$$

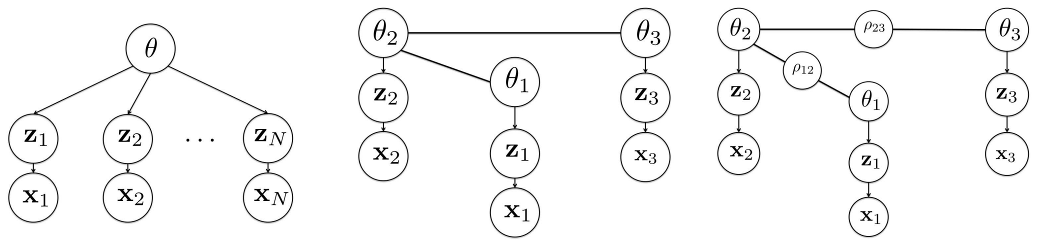
with

$$p(\mathbf{X}, \mathbf{Z} | \theta) = \prod_n p(\mathbf{x}_n, \mathbf{z}_n | \theta),$$

as depicted in Figure 1.1b. In this general model, we can find an optimal global parameter $\hat{\theta}$ (in a MAP sense) by applying standard EM learning. The EM follows a recursive two-step procedure: (a) E-step, where the posterior density $p(\mathbf{z}_n | \mathbf{x}_n, \theta)$ is estimated, and (b) M-step: parametric optimization

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}_{\mathbf{Z}|\mathbf{X}} \left[\log p(\mathbf{X}, \mathbf{Z} | \theta) \right]. \tag{3.1}$$

It is important to point out that each posterior density estimate at point n depends solely on the corresponding measurement $\mathbf{x}_n$, and does not depend on any other $\mathbf{x}_k, k \neq n$. This means that even if we partition independent measurements into arbitrary subsets, posterior density estimation is accomplished locally, within each subset. However, in the M-step all measurements $\mathbf{X}$ affect the choice of $\hat{\theta}$ because of the dependence of each term in the completed log likelihood on the same $\hat{\theta}$. This is a typical characteristic of parametric models, where the optimal parameters depend on summary data statistics.



(a) Centralized (b) Distributed (c) Decentralized

Figure 3.1: Centralized, distributed and augmented models for probabilistic PCA.

3.1.2 Distributed Setting

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected connected graph with vertices $i, j \in \mathcal{V}$ and edges $e_{ij} = (i, j) \in \mathcal{E}$ connecting the two vertices. Each i-th node is directly connected with 1-hop neighbors in $\mathcal{B}_i = \{j | e_{ij} \in \mathcal{E}\}$. Suppose the set of data samples at the i-th node is $\mathbf{X}_i = \{\mathbf{x}_{in} | n = 1, ..., N_i\}$, where $\mathbf{x}_{in} \in \mathcal{R}^D$ is n-th measurement vector and N_i is the number of samples collected in the i-th node. Likewise, we define the latent variable set for node i as $\mathbf{Z}_i = \{\mathbf{z}_{in} | n = 1, ..., N_i\}$.

As observed previously, each posterior estimation is decentralized. Learning the model parameter would be decentralized if each node had its own independent parameter θ_i. Still, the centralized model can be equivalently defined using the set of local parameters, with an additional constraint on their consensus, $\theta_1 = \theta_2 = \cdots = \theta_{|\mathcal{V}|}$. This is illustrated in Figure 3.3b where the local node models are constrained using ties defined on the underlying graph.

3.1.3 Decentralized Setting

The simple consensus tying can be more conveniently defined using a set of auxiliary variables ρ_{ij}, one for each edge e_{ij} (Figure 3.3c). This now leads to the final distributed consensus learning formulation, similar to [36]:

$$\begin{aligned} \hat{\boldsymbol{\theta}} = \underset{\{\theta_i : i \in \mathcal{V}\}}{\arg\min} \quad & -\log p(\mathbf{X} | \boldsymbol{\theta}, \mathcal{G}) \\ s.t. \quad & \theta_i = \rho_{ij}, \rho_{ij} = \theta_j, \quad i \in \mathcal{V}, j \in \mathcal{B}_i \end{aligned} \tag{3.2}$$

where we marginalized on $\mathbf{X}$. This is a constrained optimization task that can be solved in a principal manner using the Alternating Direction Method of Multipliers (ADMM) [82, 83, 15]. A brief review on ADMM is provided in Appendix A. ADMM iteratively, in a block-coordinate fashion, solves $\max_\lambda \min_\theta \mathcal{L}(\cdot)$ on the augmented Lagrangian

$$\mathcal{L}(\boldsymbol{\theta}, \rho, \lambda) = -\log p(\mathbf{X}|\theta_1, \theta_2, ..., \theta_{|\mathcal{V}|}, \mathcal{G}) + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{B}_i} \left\{\lambda_{ij1}^{\mathrm{T}}(\theta_i - \rho_{ij}) + \lambda_{ij2}^{\mathrm{T}}(\rho_{ij} - \theta_j)\right\}$$
$$+ \frac{\eta}{2} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{B}_i} \left\{||\theta_i - \rho_{ij}||^2 + ||\rho_{ij} - \theta_j||^2\right\} \tag{3.3}$$

where $\lambda_{ij1}, \lambda_{ij2}, i, j \in \mathcal{V}$ are the Lagrange multipliers, η is some positive scalar parameter and $||\cdot||$ is the induced norm. The last term (modulated by η) is not strictly necessary for consensus but introduces additional regularization. Further discussions on this term and the parameter can be found in [15] and [35]. The auxiliary ρ_{ij} plays a critical decoupling role and separate estimation of local θ_i during block-coordinate ascent/descent. This classic (first introduced in the 1970s) meta decompose algorithm can be used to devise a distributed counterpart for any centralized problem that attempts to maximize a global log likehood function over a connected network.

3.2 Example: Distributed Probabilistic Principal Component Analysis

We now apply the general distributed probabilistic learning explained above to the specific case of distributed PPCA. We first briefly review the principal component analysis (PCA) and probabilistic PCA (PPCA). Then we show how to derive distributed counterpart of the PPCA.

3.2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a classic feature selection method that finds a linear projection that maximizes the variance of projected data, called the principal component. Finding the projection for the first principal component for a given data matrix $\mathbf{X}$ is an optimization problem

$$\mathbf{w}_{(1)} = \arg\max_{\mathbf{w}} \quad \mathbf{w}^\top(\mathbf{X}\mathbf{X}^\top)\mathbf{w} \tag{3.4}$$

$$s.t. \quad \|\mathbf{w}\| = 1. \tag{3.5}$$

All principal components can be found using eigendecomposition as $\mathbf{X} = \mathbf{W}\mathbf{Z}$ where columns of $\mathbf{W}$ are eigenvectors of $\mathbf{X}\mathbf{X}^\top$. Thus, PCA can be interpreted as a matrix decomposition method that decomposes a given matrix $\mathbf{X}$ into orthonormal $\mathbf{W}$ and $\mathbf{Z}$. It is easy to see that the singular value decomposition (SVD) is a solution to this PCA problem. To see this, let $\mathbf{X} = \mathbf{W}\mathbf{\Sigma}\mathbf{V}^\top$ be the singular value decomposition result of $\mathbf{X}$. Then,

$$\mathbf{X}\mathbf{X}^\top = \mathbf{W}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}\mathbf{W}^\top = \mathbf{W}\mathbf{\Sigma}^2\mathbf{W}^\top \tag{3.6}$$

thus columns of $\mathbf{W}$ are eigenvectors of $\mathbf{X}\mathbf{X}^\top$. Now,

$$\mathbf{Z} = \mathbf{W}^\top\mathbf{X} = \mathbf{W}^\top\mathbf{W}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{\Sigma}\mathbf{V}^\top. \tag{3.7}$$

We will use the centralized SVD result on input $\mathbf{X}$ as a baseline PCA in the experiments.

3.2.2 Probabilistic PCA (PPCA)

Traditional centralized formulation of probabilistic PCA (PPCA) [11] assumes that latent variable $\mathbf{z}_{in} \sim \mathcal{N}(\mathbf{z}_{in}|0, \mathbf{I})$, with a generative relation

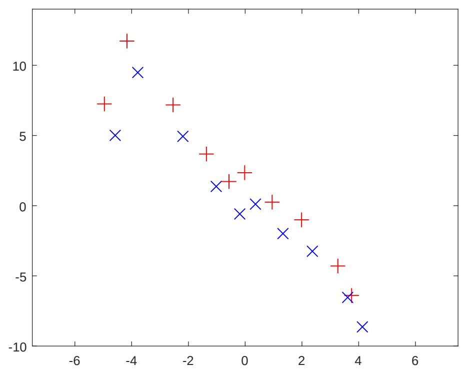
$$\mathbf{x}_{in} = \mathbf{W}_i\mathbf{z}_{in} + \boldsymbol{\mu}_i + \boldsymbol{\epsilon}_i, \tag{3.8}$$

where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{\epsilon}_i|0, a_i^{-1}\mathbf{I})$ and a_i is the noise precision. Inference then yields

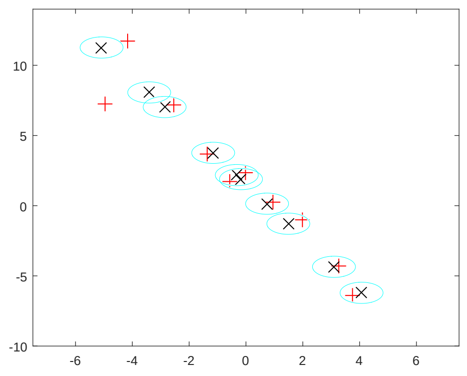
$$p(\mathbf{z}_{in}|\mathbf{x}_{in}) = \mathcal{N}(\mathbf{z}_{in}|\mathbf{L}_i^{-1}\mathbf{W}_i^{\mathrm{T}}(\mathbf{x}_{in} - \boldsymbol{\mu}_i), a_i^{-1}\mathbf{L}_i^{-1}), \tag{3.9}$$

where $\mathbf{L}_i = \mathbf{W}_i^{\mathrm{T}}\mathbf{W}_i + a_i^{-1}\mathbf{I}$. We can find optimal parameters $\mathbf{W}_i, \boldsymbol{\mu_i}, a_i$ by determining the maximum likelihood estimates of the marginal data likelihood, or by applying the EM algorithm on expected complete data log likelihood with respect to the posterior density $p(\mathbf{Z}_i|\mathbf{X}_i)$.

Benefits of this probabilistic treatment are threefold: (a) We can derive an EM algorithm for the PCA that is computationally efficient, if we are interested in a few



(a) PCA



(b) PPCA

Figure 3.2: Comparison between PCA and PPCA. Red cross denotes input data, blue cross denotes PCA reprojection, black cross denotes PPCA reprojection, cyan circle denotes estimated one standard deviation from mean estimate of reprojection.

leading eigenvectors (i.e., latent dim << observation dim), (b) We can naturally obtain estimates of latent space, and (c) In combination with EM, it can deal with missing values, which will be discussed more in detail in Section 3.2.4. Figure 3.2, demonstrates this benefit at a glance. Here, given two-dimensional data $\mathbf{X}$, we want to find the one-dimensional prinxipal axis. With PCA, we can find a rough linear subspace, as shown in the center figure. However, PPCA can provide additional information on input variance, thus we can actually estimate how far each input data point is from its corresponding subspace projection, as shown in the right figure. One can show that PCA is a special case of PPCA, when the additive observation noise variance is zero.

3.2.3 Distributed PPCA (D-PPCA)

The distributed algorithm developed in Section 3.1.3 can be directly applied to this PPCA model. The basic idea is to assign each subset of samples as evidence for the local generative models with parameters $\mathbf{W}_i, \boldsymbol{\mu}_i, a_i^{-1}$. The inference is accomplished locally in each node. The local parameter estimates are then computed using the consensus updates that combine local summary data statistics with the information about the model conveyed through neighboring network nodes. Below, we outline specific details of this approach.

Let $\mathbf{\Theta}_i = \{\mathbf{W}_i, \boldsymbol{\mu}_i, a_i\}$ be the set of parameters for each node i. The global constrained consensus optimization now becomes

$$\begin{aligned}
&\underset{\{\mathbf{W}_i,\boldsymbol{\mu}_i,a_i:i\in\mathcal{V}\}}{\arg\min} && -F(\mathbf{\Theta}_i)\\
& && \mathbf{W}_i = \boldsymbol{\rho}_{ij}, \quad \boldsymbol{\rho}_{ij} = \mathbf{W}_j, \quad i \in \mathcal{V}, j \in \mathcal{B}_i,\\
&s.t. && \boldsymbol{\mu}_i = \boldsymbol{\phi}_{ij}, \quad \boldsymbol{\phi}_{ij} = \boldsymbol{\mu}_j, \quad i \in \mathcal{V}, j \in \mathcal{B}_i,\\
& && a_i = \psi_{ij}, \quad \psi_{ij} = a_j, \quad i \in \mathcal{V}, j \in \mathcal{B}_i
\end{aligned}$$

where $F(\mathbf{\Theta}_i) = \sum_{n=1}^{N_i} \log p(\mathbf{x}_{in}|\mathbf{W}_i, \boldsymbol{\mu}_i, a_i^{-1})$. The augmented Lagrangian is

$$\begin{aligned}
\mathcal{L}(\mathbf{\Phi}_i) = &-F(\mathbf{\Theta}_i)\\
&+ \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i} \left(\boldsymbol{\lambda}_{ij1}^{\mathrm{T}}(\mathbf{W}_i - \boldsymbol{\rho}_{ij}) + \boldsymbol{\lambda}_{ij2}^{\mathrm{T}}(\boldsymbol{\rho}_{ij} - \mathbf{W}_j)\right)\\
&+ \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i} \left(\boldsymbol{\gamma}_{ij1}^{\mathrm{T}}(\boldsymbol{\mu}_i - \boldsymbol{\phi}_{ij}) + \boldsymbol{\gamma}_{ij2}^{\mathrm{T}}(\boldsymbol{\phi}_{ij} - \boldsymbol{\mu}_j)\right)\\
&+ \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i} \left(\beta_{ij1}(a_i - \psi_{ij}) + \beta_{ij2}(\psi_{ij} - a_j)\right)\\
&+ \frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i} (||\mathbf{W}_i - \boldsymbol{\rho}_{ij}||^2 + ||\boldsymbol{\rho}_{ij} - \mathbf{W}_j||^2)\\
&+ \frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i} (||\boldsymbol{\mu}_i - \boldsymbol{\phi}_{ij}||^2 + ||\boldsymbol{\phi}_{ij} - \boldsymbol{\mu}_j||^2)\\
&+ \frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i} ((a_i - \psi_{ij})^2 + (\psi_{ij} - a_j)^2) \qquad (3.10)
\end{aligned}$$

where $\mathbf{\Phi}_i = \{\mathbf{W}_i, \boldsymbol{\mu}_i, a_i, \boldsymbol{\rho}_{ij}, \boldsymbol{\phi}_{ij}, \psi_{ij}; i \in \mathcal{V}, j \in \mathcal{B}_i\}$ and $\{\boldsymbol{\lambda}_{ijk}\}, \{\boldsymbol{\gamma}_{ijk}\}, \{\beta_{ijk}\}$ with $k = 1, 2$ are the Lagrange multipliers. The scalar value η gives us control over the convergence speed of the algorithm. With reasonably large positive η, the overall optimization converges fairly quickly [36]. We will explore the converging behaviour with respect to various η in synthetic data experiments.

Similar to a standard EM approach, we minimize the upper bound of $\mathcal{L}(\mathbf{\Phi}_i)$. Exploiting the posterior density in (3.9), we compute the expected mean and variance of latent variables in each node as

$$\mathbb{E}[\mathbf{z}_{in}] = \mathbf{L}_i^{-1}\mathbf{W}_i^{\mathrm{T}}(\mathbf{x}_{in} - \boldsymbol{\mu}_i), \qquad (3.11)$$

$$\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^{\mathrm{T}}] = a_i^{-1}\mathbf{L}_i^{-1} + \mathbb{E}[\mathbf{z}_{in}]\mathbb{E}[\mathbf{z}_{in}]^{\mathrm{T}}. \qquad (3.12)$$

Maximization of the completed likelihood Lagrangian derived from (3.10) yields

$$\begin{aligned} \mathbf{W}_i^{(t+1)} &= \left\{ a_i \sum_{n=1}^{N_i} (\mathbf{x}_{in} - \boldsymbol{\mu}_i)\mathbb{E}[\mathbf{z}_{in}]^{\mathrm{T}} - 2\boldsymbol{\lambda}_i^{(t)} + \eta \sum_{j \in B_i} \left(\mathbf{W}_i^{(t)} + \mathbf{W}_j^{(t)} \right) \right\} \\ &\quad \cdot \left(a_i \sum_{n=1}^{N_i} \mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^{\mathrm{T}}] + 2\eta|\mathcal{B}_i|\mathbf{I} \right)^{-1}, \qquad (3.13) \\ \boldsymbol{\mu}_i^{(t+1)} &= \left\{ a_i \sum_{n=1}^{N_i} \left(\mathbf{x}_{in} - \mathbf{W}_i \mathbb{E}[\mathbf{z}_{in}] \right) - 2\boldsymbol{\gamma}_i^{(t)} + \eta \sum_{j \in B_i} \left(\boldsymbol{\mu}_i^{(t)} + \boldsymbol{\mu}_j^{(t)}) \right) \right\} \\ &\quad \cdot \left(N_i a_i + 2\eta|\mathcal{B}_i| \right)^{-1}, \qquad (3.14) \\ \boldsymbol{\lambda}_i^{(t+1)} &= \boldsymbol{\lambda}_i^{(t)} + \frac{\eta}{2} \sum_{j \in \mathcal{B}_i} \left\{ \mathbf{W}_i^{(t+1)} - \mathbf{W}_j^{(t+1)} \right\}, \qquad (3.15) \\ \boldsymbol{\gamma}_i^{(t+1)} &= \boldsymbol{\gamma}_i^{(t)} + \frac{\eta}{2} \sum_{j \in \mathcal{B}_i} \left\{ \boldsymbol{\mu}_i^{(t+1)} - \boldsymbol{\mu}_j^{(t+1)} \right\}, \qquad (3.16) \\ \beta_i^{(t+1)} &= \beta_i^{(t)} + \frac{\eta}{2} \sum_{j \in \mathcal{B}_i} \left\{ a_i^{(t+1)} - a_j^{(t+1)} \right\}. \qquad (3.17) \end{aligned}$$

For a_i, we solve the quadratic equation

$$\begin{aligned} 0 = &-\frac{N_i D}{2} + 2\eta|\mathcal{B}_i| {a_i^{(t+1)}}^2 \\ &+ a_i^{(t+1)} \cdot \Bigg\{ 2\beta_i^{(t)} - \eta \sum_{j \in B_i} \left(a_i^{(t)} + a_j^{(t)} \right) - \sum_{n=1}^{N_i} \mathbb{E}[\mathbf{z}_{in}]^{\mathrm{T}} \mathbf{W}_i^{\mathrm{T}} (\mathbf{x}_{in} - \boldsymbol{\mu}_i) \\ &\qquad + \frac{1}{2} \sum_{n=1}^{N_i} \left\{ ||\mathbf{x}_{in} - \boldsymbol{\mu}_i||^2 + tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^{\mathrm{T}}] \mathbf{W}_i^{\mathrm{T}} \mathbf{W}_i \right] \right\} \Bigg\}. \qquad (3.18) \end{aligned}$$

The overall distributed EM algorithm for D-PPCA is summarized in Algorithm 1. Full derivation can be found in Appendix B.

3.2.4 Dealing with Missing Values in Input Data

Traditional PPCA is an effective tool for dealing with data missing-at-random (MAR) in traditional PCA [84]. While more sophisticated methods including variational approximations (cf. [84]) are possible, direct use of PPCA is often sufficient in practice. Hence, we adopt D-PPCA as a method to deal with missing data in a distributed consensus setting.

Generalization to missing data D-PPCA from D-PPCA is straightforward and follows [84]. From the perspective of ADMM-based learning, the only modifications come

Algorithm 1 Distributed Probabilistic PCA (D-PPCA)

```
Require: For every node $i$ initialize $\mathbf{W}_i^{(0)}, \boldsymbol{\mu}_i^{(0)}, a_i^{(0)}$ randomly and set Lagrange multipliers for the parameters to zero, i.e., $\boldsymbol{\lambda}_i^{(0)} = \mathbf{0}, \boldsymbol{\gamma}_i^{(0)} = \mathbf{0}, \beta_i^{(0)} = 0$.
  for $t = 0, 1, 2, ...$ until convergence do
    for all $i \in V$ do
      [E-step] Compute $\mathbb{E}[\mathbf{z}_{in}]$ and $\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^{\mathrm{T}}]$ via traditional inference.
      [M-step] Compute $\mathbf{W}_i^{(t+1)}$, $\boldsymbol{\mu}_i^{(t+1)}$, $a_i^{(t+1)}$, via (3.13), (3.14) and (3.18).
    end for
    for all $i \in V$ do
      Broadcast $\mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}$, and $a_i^{(t+1)}$ to all neighbors of $i \in \mathcal{B}_i$.
    end for
    for all $i \in V$ do
      Compute $\boldsymbol{\lambda}_i^{(t+1)}$, $\boldsymbol{\gamma}_i^{(t+1)}$, and $\beta_i^{(t+1)}$ via (3.15), (3.16) and (3.17)
    end for
  end for
```

in the form of adjusted terms for local data summaries. For instance, in (B.4) the data summary term

$$\sum_{n=1}^{N_i} (\mathbf{x}_{in} - \mathbf{W}_i \mathbb{E}[\mathbf{z}_{in}]) \tag{3.19}$$

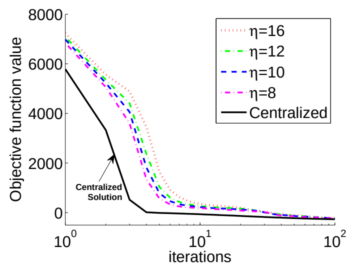
becomes

$$\sum_{n \in O_{i,f}} x_{i,n,f} - \mathbf{w}_{i,f}^T \mathbb{E}[\mathbf{z}_{in}], \tag{3.20}$$

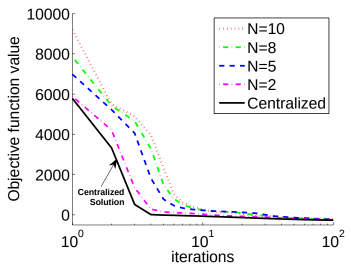
where $f = 1, \ldots, D$ is the index of feature, $O_{i,f}$ is the set of samples in node i that have the feature f present, $x_{i,n,f}$ is the value of the present feature, and $\mathbf{w}_{i,f}^T$ is the f-th row of matrix $\mathbf{W}_i$. Similar expressions can be derived for other local parameters. Note that (3.15-3.17) incur no changes.

3.2.5 Evaluation

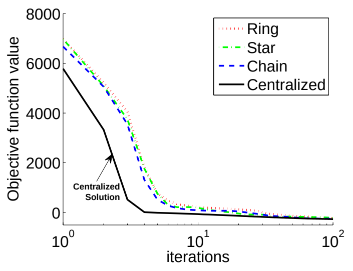
We first demonstrate the empirical convergence properties of the D-PPCA. Note that the general convergence properties are implied by the Augmented Lagrangian optimization algorithm. Additionally, in a distributed network setting the convergence will depend on the connectivity structure of the network, which in turn depends on the spectral properties of its graph Laplacian. We generated 100 50-dimensional random samples from $\mathcal{N}(0, 0.2 \cdot \mathbf{I})$. We assigned 20 samples equally to each node in a 5-node network connected with ring topology to find a 5-dimensional subspace. Our convergence



(a) Impact of η



(b) Impact of number of nodes



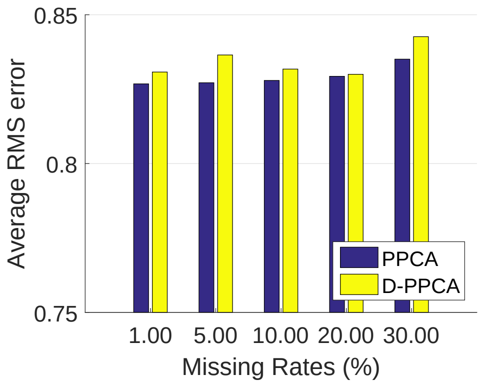
(c) Impact of topologies

Figure 3.3: Convergence trends of D-PPCA.

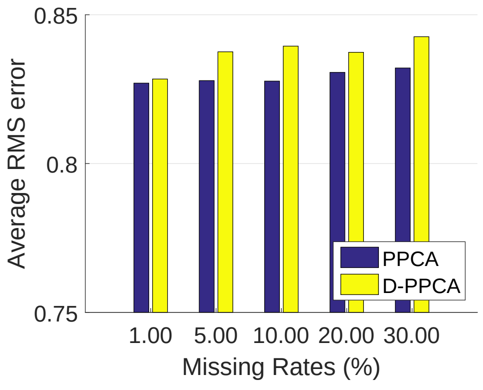
criterion is the relative change in objective of (B.1) and we stop when it is smaller than 10^{-5}. In real settings, one can monitor local parameter updates instead. We initialized parameters with random values from a uniform distribution. Alternative choices of starting points may lead to faster convergence. If not explicitly mentioned otherwise, all our results are averaged over 20 independent random initializations.

Figure 3.3a shows the convergence curve of D-PPCA for various η values. As one can easily see, all η values lead to convergence within 10^2 iterations. Moreover, the value they converge to is equivalent to a centralized solution, meaning we can achieve the same global solution using the distributed algorithm. This behavior matches results reported in [35]. Figure 3.3b shows convergence curves as a function of the number of nodes in a network. In all cases, D-PPCA successfully converged within 10^2 iterations. Similar trends were observed with networks of more than 10 nodes. We also conducted experiments to test the effects of network topology on the parameter convergence. Figure 3.3c depicts the results for three simple network types. In all cases we considered, D-PPCA reached near the stationary point within only 10 iterations, regardless of any of the aforementioned factors.

One of the main benefits of employing probabilistic formulation of PCA is the flexibility of allowing missing values. Here, we consider two possibilities of missing values; the case when values are missing at random (MAR) and the case when values are missing not at random (MNAR). In [84], it has been shown that probabilistic formulations of PCA can deal with missing values, doing this particularly well in the MAR setting.



(a) Missing At Random



(b) Missing Not At Random

Figure 3.4: Average root mean squared error of reconstructions based on PPCA and D-PPCA results.

The same conclusion holds for D-PPCA. To demonstrate, we prepared a similar dataset with 500 samples, each with 20-dimensional input data to find a 5-dimensional subspace. Input variance is the same as before (0.2). The network has 5 nodes connected as a ring shape. We generated a band matrix to simulate the MNAR case, i.e., we removed the given ratio of off-diagonal feature values from the input data matrix. As shown in Figure 3.4a, D-PPCA can effectively reconstruct the original measurement comparable to its centralized counterpart under different amounts of missing values. This fact also holds for the MNAR case, although the error tends to be slightly larger than in the MAR case, as shown in Figure 3.4b. The difference between centralized and distributed is due to outliers over different initializations, and we will see how Bayesian approach can mitigate this issue in the next chapter.

3.3 Application: Distributed Computer Vision

We now show that the modified D-PPCA can be used as an effective framework for distributed affine SfM. We first show results in a controlled environment with synthetic data, and then report results on data from real video sequences. We assume that correspondences across frames and cameras are known. For the missing values of the MNAR case, we either used the actual occlusions to induce missing points, or simulated consistently missing points over several frames.

3.3.1 Affine Structure from Motion

In this section, we consider a specific formulation of the modified distributed probabilistic PCA for application in the affine Structure from Motion problem. Structure from Motion (SfM) is a computer vision problem, involving the desire to reconstruct the 3D structure of the scene and camera motion simultaneously, given two or more camera views observing a common scene [85]. There are two assumptions in this problem: (a) 2D points observed from cameras are of a rigid object, and (b) points' correspondence across the cameras is known. The affine SfM, adds an additional assumption that the 3D-2D relationship is always an affine transformation. It is possible to relax these constraints, but we maintain these assumptions in order to focus on our purpose of demonstrating D-PPCA's ability to decentralized computation.

Formally, our goal is to estimate the 3D location of N points on a rigid object based on corresponding 2-D points observed from multiple cameras (or views). The dimension D of our measurement matrix is thus twice the number of frames each camera observed. A simple and effective way to solve this problem is the factorization method [85]. Given a 2D (image coordinate) measurement matrix $\mathbf{X}$, of size $2 \cdot \#frames \times \#points$, the matrix is factorized into a $2 \cdot \#frames \times 3$ motion matrix $\mathbf{M}$ and the $3 \times \#points$ 3D structure matrix $\mathbf{S}$. In the centralized setting this can be easily computed using SVD on $\mathbf{X}$. Equivalently, the estimates of $\mathbf{M}$ and $\mathbf{S}$ can be found using inference and learning in a centralized PPCA, where $\mathbf{M}$ is treated as the PPCA parameter and $\mathbf{S}$ is the latent structure. There we obtain additional estimates of the variance of structure $\mathbf{S}$, which are not immediately available from the factorization approach, although, they can be found.

However, the above defined $(2 \cdot \#frames \times \#points)$ data structure of $\mathbf{X}$ is not amenable to distribution of different views (cameras, nodes), as considered in Section 3.2.3 of D-PPCA. Namely, D-PPCA assumes that the distribution is accomplished by splitting the data matrix $\mathbf{X}$ into sets of non-overlapping columns, one for each node. Here, however, we seek to distribute the rows of matrix $\mathbf{X}$, i.e., a set of (subsequent) frames is to be assigned to each node/camera.

Hence, to apply the D-PPCA framework to SfM we need to swap the role of rows and columns, i.e., consider modeling of $\mathbf{X}^\top$. This, subsequently, means that the 3D scene structure (which is to be shared across all nodes in the network) will be treated as the D-PPCA *parameter*. The latent D-PPCA variables will model the unknown and uncertain motion of each camera (and/or object in its view).

Specifically, we will consider the model

$$\mathbf{X}_i^\top = \mathbf{W} \cdot \mathbf{Z}_i + \mathbf{E}_i \tag{3.21}$$

where $\mathbf{X}_i$ is the matrix of image coordinates of all points in node (camera) i of size $\#points \times 2 \cdot \#frames$ in node i, $\mathbf{W}$ is the $\#points \times 3$ 3D structure (D-PPCA parameter) matrix and $\mathbf{Z}_i$ is the $3 \times 2 \cdot \#frames$ motion matrix of node i.

One should note that we have implicitly assumed, in a standard D-PPCA manner, that each column of $\mathbf{Z}_i$ is i.i.d. and distributed as $\mathcal{N}(0, \mathbf{I})$. However, each pair of subsequent $\mathbf{Z}_i$ columns represents one 3×2 affine motion matrix. While those columns are not truly independent, our experiments (as demonstrated in Sections 3.3.2 and 3.3.3) show that this assumption is not detrimental in practice. The final task is simply following the same process we carried out to derive D-PPCA.

Missing data in SfM will be handled using the formalism presented in Section 3.2.4. Strictly speaking, the model of data missing-at-random is not always applicable to SfM. The reason is that occlusions, the main source of missing data, cannot be treated as a random process. Instead, this setting corresponds to data missing-not-at-random [84] (MNAR). If treated blindly, this may introduce bias in the estimated models. However, as we demonstrate in experiments, this assumption does not adversely affect SfM when the number of missing points is within a reasonable range.

3.3.2 Results on Synthetic Data

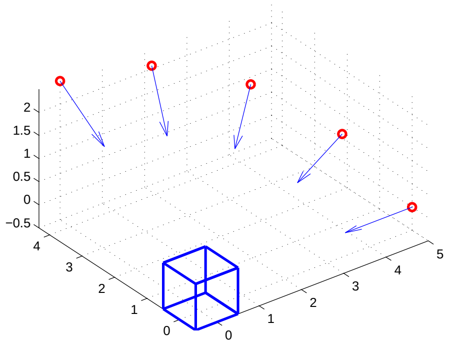
We first generated synthetic data with a rotating unit cube and 5 cameras facing the cube in a 3D space, similar to synthetic experiments in [60]. The cube is centered at the origin of the space and rotates 30° counterclockwise. We extracted 8 cube points projected on each camera view every 6°, i.e., each camera observed 5 frames.

Cameras are placed on a skewed plane, making an elevation along the z-axis, as shown in Figure 3.5a. For all synthetic and real SfM experiments, we picked $\eta = 10$ and initialized $\mathbf{W}_i$ matrix with feature point coordinates of the first frame visible in the i-th camera with some small amount of noise. The convergence criterion for D-PPCA for SfM was set as 10^{-3} relative error. To measure the performance, we computed maximum subspace angle between the ground truth 3D coordinates and our estimated 3D structure matrix. For comparison, we conducted traditional SVD-based SfM on the same data. In the noise-free case, D-PPCA for SfM always yielded the same performance as SVD-based SfM with near $0°$.

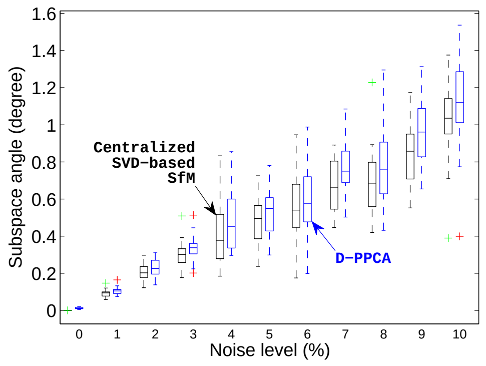
We also tested D-PPCA for SfM with noisy and missing-value cases. First, we generated 20 independent samples of all 25 frames with 10 different noise levels. Then we ran D-PPCA 20 times on each of the independent samples, and averaged the final structure estimates. As Figure 3.5b shows, we found that D-PPCA for SfM is fairly robust to noise and tends to stabilize even as the noise level increases. The mean subspace angle tends to be slightly larger than that estimated by the centralized SVD SfM, however both reside within the overlapping confidence intervals. Considering MAR missing values, we obtained $1.66°$ for 20% missing points averaged over 10 different missing point samples. In the MNAR case with actual occlusions considered, D-PPCA yielded, a relatively larger, $20°$ error. Intuitively, this is because the missing points in the scene are naturally not random. However, we argue that D-PPCA can still handle missing points, given the evidence in the next section.

3.3.3 Results on Real Data

For real data experiements, we first applied D-PPCA for SfM on the Caltech 3D Objects on Turntable dataset [64]. The dataset provides various objects rotating on a turntable under different lighting conditions. The views of most objects were taken every $5°$ which make it challenging to extract feature points with correspondence across frames. Instead, we used a subset of the dataset which provides views taken every degree. This subset contains images of 5 objects. To simulate multiple cameras, we adopted a setting similar to that of [60]. We initially extracted the first $30°$ images of each



(a) Camera Setting



(b) Subspace Angle vs. Ground Truth

Figure 3.5: Rotating unit cube with multiple cameras. Red circles are camera locations and blue arrows indicate each camera's facing direction. Green and red crosses in the right plot are outliers for centralized SVD-based SfM and D-PPCA for SfM, respectively.

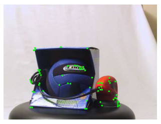
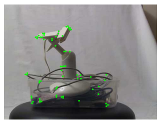
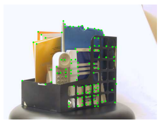
object. We then used KLT [86] implementation in Voodoo Camera Tracker[1] to extract feature points with correspondence. Lastly, we sequentially and equally partitioned the 30 images into 5 nodes to simulate 5 cameras. Thus, each camera observes 6 frames. Table 3.1 shows the 5 objects and statistics of feature points we extracted from the objects. We used $\eta = 10$ and convergence criterion 10^{-3}. Due to the lack of the ground truth 3D coordinates, we compared the subspace angles between the structure inferred using the traditional centralized SVD-based SfM and the D-PPCA-based SfM. Results are shown in Table 3.1 as the mean and variance of 20 independent runs. 10% MAR and MNAR results are also provided in the table.

Experimental results indicate the existance of differences between the reconstructions obtained by the centralized factorization approach, and that of D-PPCA. However, the differences are small, depend on the object in question, and almost always include, within their confidence, the factorization result. Qualitative examination reveals no noticable differences. Moreover, reprojecting back to the camera coordinate space resulted in close matching with the tracked feature points, as shown in videos provided in supplementary materials.

We also tested the utility of D-PPCA for SfM on the Hopkins155 dataset [66]. We

[1]http://www.viscoda.com/index.php/en/products/non-commercial/voodoo-camera-tracker

Table 3.1: Caltech 3D Objects on Turntable dataset statistics and quantitative results. Green dots indicate feature points tracked with correspondence across all 30 frames. All results ran 20 independent initializations. MAR results provide variances over both various initializations and missing value settings. Numbers are subspace angles between the two denoted methods, thus the smaller the better.

Object	BallSander	BoxStuff	Rooster	Standing	StorageBin
					
# Points	62	67	189	310	102
# Frames	30	30	30	30	30
Centralized SVD SfM vs. D-PPCA (degree)					
Mean	1.4848	1.4397	1.4767	2.6221	0.4463
Variance	0.4159	0.4567	0.9448	1.6924	1.2002
Fully observable centralized PPCA SfM vs. D-PPCA with MAR (degree)					
Mean	6.2991	2.1556	5.2506	7.6492	2.8358
Var.(init)	4.3562	0.1351	3.8810	6.6424	1.3591
Var.(miss)	0.5729	0.0161	0.1755	0.7603	0.0444
Fully observable centralized PPCA SfM vs. D-PPCA with MNAR (degree)					
Mean	3.1405	6.4664	5.8027	9.2661	3.7965
Variance	0.0124	3.1955	2.4333	2.9720	0.0089

adopted a virtually identical experimental setting as in [60]. We collected 135 single-object sequences containing image coordinates of points, and we simulated a multi-camera setting by partitioning the frames sequentially and almost equally for 5 nodes, and the network was connected using ring topology. Again, we computed maximum subspace angle between centralized SVD-based SfM and distributed D-PPCA for SfM. We chose the convergence criterion as 10^{-3}. The average maximum subspace angle between D-PPCA for SfM and SVD-based SfM for all objects was 3.97° with variance 7.06.

However, looking into the result more carefully, we found that even with substantially larger subspace angle, 3D structure estimates were similar to that of SVD-based SfM, only with an orthogonal ambiguity issue. Moreover, more than 53% of all objects yielded a subspace angle below 1°, 77% of them below 5° and more than 94% were less than 15°. With 10% MAR, we obtained the mean 20.07° with variance 27.94° with approximately 18% of them below 1°, 56% of them below 5° and more than 70% of them less than the mean. We could not perform MNAR experiments on Hopkins as the ground truth occlusion information is not provided with the dataset.

3.4 Summary

In this chapter we introduced a general approach for learning parameters of traditional centralized probabilistic models, such as PPCA, in a distributed setting. Our synthetic data experiments showed that the proposed algorithm is robust to choices of initial parameters and, more importantly, is not adversely affected by variations in network size, topology or missing values. In the SfM problems, the algorithm can be effectively used to distribute computation of 3D structure and motion in camera networks, while retaining the probabilistic nature of the original model.

Despite its promising performance, D-PPCA for SfM exhibits some limitations. In particular, we assume the independence of the affine motion matrix parameters in (3.21). The assumption is clearly inconsistent with the modeling of motion on the SE(3) manifold. However, our experiments demonstrate that, in practice, this violation is not crucial. This shortcoming can be amended in one of several possible ways. One can reduce the i.i.d. assumption of individual samples to that of subsequent columns (i.e., full 3x2 motion matrices). Our additional experiments, not reported here, indicate no discernible utility of this approach. A more principled approach would be to define priors for motion matrices compatible with SE(3), using e.g., [87]. While appealing, the priors would render the overall model non-linear, and would require additional algorithmic considerations, perhaps in the spirit of [60].

Chapter 4

Extensions of Distributed Probabilistic Learning

In this chapter, we introduce two effective extensions of the distributed learning framework we proposed in the previous chapter. First, we investigate how our framework can be extended to adapt to dynamically changing input data, i.e., in an online learning setting. This is a further generalization of the distributed model estimation problem we discussed in the previous chapter. Next, we investigate how one can improve the speed of convergence of the underlying ADMM optimization algorithm that can benefit not only our framework but also, potentially, other ADMM-based methods.

4.1 Online Distributed Probabilistic Learning

Online learning in machine learning means that the model can be adaptively updated based on dynamically changing input data. The change can be either in the form of streaming (i.e., retrieve small chunks of data at a time) or in the form of a batch (i.e., incrementally add new data, but learning is done from scratch everytime). The ability to handle such data in an online fashion is crucial in large data analysis, as the data can easily outfit the physical memory size. Therefore, it is one of the essential aspects desirable for any distributed learning methods.

Our consensus-based distributed probabilistic learning framework can be extended to handle dynamic input data by reformulating it as a Bayesian learning framework. There are a number of benefits of the employment of the Bayesian framework. First, it can naturally learn and update the probabilistic model in an online fashion. In addition, unlike the maximum likelihood estimation approaches we performed in the previous chapter, we can obtain the full posterior distributions of parameters which gives richer information of the learned model. Finally, it can help avoid overfitting.

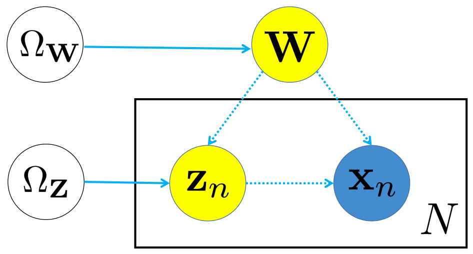


Figure 4.1: A graphical representation of the model of Eq. 4.1. Blue-shaded circle denotes observation.

4.1.1 Distributed Bayesian Learning Model

We first explain a general parametric Bayesian model in a centralized setting. Then, we extend it into a distributed form.

Centralized Setting. We consider a data set $\mathbf{X}$ of N observed D-dimensional column vectors $\mathbf{X} = \{\mathbf{x}_n \in \mathbb{R}^D\}, n = 1..N$ with the corresponding *local* latent variables $\mathbf{Z} = \{\mathbf{z}_n \in \mathbb{R}^M\}$, a *global* latent variable $\mathbf{W} \in \mathbb{R}^{D\times M}$, and a set of fixed parameters $\Omega = [\Omega_{\mathbf{Z}}, \Omega_{\mathbf{W}}]$. We consider the family of probabilistic models as the joint distribution of the observations and both the global and local variables. A key assumption here is that the joint distribution can be fully factorized into a global term and a product of local terms as

$$p(\mathbf{X}, \mathbf{Z}, \mathbf{W}|\Omega) = p(\mathbf{W}|\Omega_{\mathbf{W}}) \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{W}) p(\mathbf{z}_n|\Omega_{\mathbf{Z}}). \tag{4.1}$$

Figure 4.1 depicts the graphical representation of the family of probabilistic models of this type. The goal is to compute the posterior distribution of the latent variables, $p(\mathbf{W}, \mathbf{Z}|\mathbf{X}, \Omega)$. It is often mathematically convenient to assume that the conditional distributions of a latent variable, given the observation and the other variables, follow an exponential family distribution. Specifically, let $\Theta_{\mathbf{W}} = \{\mathbf{X}, \mathbf{Z}, \Omega_{\mathbf{W}}\}$ and $\Theta_{\mathbf{z}_n} = \{\mathbf{x}_n, \mathbf{W}, \Omega_{\mathbf{Z}}\}$. Then the conditional distributions can be written as

$$p(\mathbf{W}|\mathbf{X}, \mathbf{Z}, \Omega_{\mathbf{W}}) = h(\mathbf{W}) \exp\left\{\Psi\left(\Theta_{\mathbf{W}}\right)^{\top} \mathcal{T}(\mathbf{W}) - \mathcal{A}_{\mathbf{W}}\left(\Psi\left(\Theta_{\mathbf{W}}\right)\right)\right\}, \tag{4.2}$$

$$p(\mathbf{Z}|\mathbf{X}, \mathbf{W}, \Omega_{\mathbf{Z}}) = \prod_{n=1}^{N} h(\mathbf{z}_n) \exp\left\{\Psi\left(\Theta_{\mathbf{z}_n}\right)^{\top} \mathcal{T}(\mathbf{z}_n) - \mathcal{A}_{\mathbf{Z}}\left(\Psi\left(\Theta_{\mathbf{z}_n}\right)\right)\right\}, \tag{4.3}$$

where $h(.)$ is the *base measure*, $\mathcal{A}(.)$ denotes the *log partition function*, and $\Psi(\cdot)$ is the *natural parameter*, and $\mathcal{T}(\cdot)$ denotes the *sufficient statistics*. Many well known statistical models make similar assumptions. For example, Bayesian PCA and Bayesian Mixture of PCA [88], Latent Dirichlet Allocation [89], Bayesian Gaussian Mixture model [90], and Hidden Markov model [91, 92] belong to this class of models. Computing the exact posterior distributions of the latent variables, given observations, are often intractable even with the exponential family assumptions. A typical workaround is to use approximate inference algorithms.

Variational Inference (VI). VI is an inference method that approximates the true distribution over the latent variables with a simpler distribution indexed by a set of free parameters that is the closest in Kullback-Leibler (KL) divergence to the true posterior distribution [93, 94]. There are two merits of VI over other approximation methods such as sampling. First, with appropriate choices of the simpler distributions, we can derive the closed-form solution for the posterior computation that could run more quickly. Second, it is unclear how to sample in a decentralized setting. There are some recent works such as parallel stochastic Markov Chain Monte Carlo [95], but sampling in a decentralized way remains an open problem that is not trivial to solve.

In VI, we first pick a family of distributions over the latent variables with its own parameters, e.g.,

$$p(\mathbf{W}|\mathbf{X},\mathbf{Z},\Omega_{\mathbf{W}}) \approx q(\mathbf{W}|\lambda_{\mathbf{W}}) \tag{4.4}$$

where the new parameter $\lambda_{\mathbf{W}}$ is called the *variational parameter* and the $q(\cdot)$ is termed *variational distribution*. The goal is to find the best set of parameters that makes the chosen $q(\cdot)$ to be as close as possible to the true density. For the *closeness* measure between the two distributions, we use the KL divergence [96, 97]. Using the above example, the KL divergence of $p(\mathbf{W}|\mathbf{X},\mathbf{Z},\Omega_{\mathbf{W}})$ from $q(\mathbf{W}|\lambda_{\mathbf{W}})$ is defined as

$$\mathrm{KL}(q(\mathbf{W}|\lambda_{\mathbf{W}})\|p(\mathbf{W}|\mathbf{X},\mathbf{Z},\Omega_{\mathbf{W}})) = \mathbb{E}_q\left[\log\frac{q(\mathbf{W}|\lambda_{\mathbf{W}})}{p(\mathbf{W}|\mathbf{X},\mathbf{Z},\Omega_{\mathbf{W}})}\right]. \tag{4.5}$$

However, exact minimization of the KL divergence is not easy, so we use a function that is equal to it up to a constant. To see this, we can rewrite (4.5) as

$$\mathrm{KL}(q(\mathbf{W}|\lambda_{\mathbf{W}})\|p(\mathbf{W}|\mathbf{X})) = \mathbb{E}_q\left[\log\frac{q(\mathbf{W}|\lambda_{\mathbf{W}})}{p(\mathbf{W},\mathbf{X})}\right] + \log p(\mathbf{X}) \tag{4.6}$$

where we omitted given $\mathbf{Z}, \Omega_{\mathbf{W}}$ for notational brevity and focus on $\mathbf{W}, \mathbf{X}$. Equivalently,

$$\log p(\mathbf{X}) = \mathrm{KL}(q(\mathbf{W}|\lambda_{\mathbf{W}})\|p(\mathbf{W}|\mathbf{X})) - \mathbb{E}_q\left[\log\frac{q(\mathbf{W}|\lambda_{\mathbf{W}})}{p(\mathbf{W},\mathbf{X})}\right]. \tag{4.7}$$

The second term on the right hand side is called the negative evidence lower bound (ELBO) or negative variational free energy because it is the negative of the lower bounding of the marginal evidence $\log p(\mathbf{X})$ as one can see from the derivation:

$$\begin{aligned}\log p(\mathbf{X}) &= \log\int_{\mathbf{W}} p(\mathbf{X},\mathbf{W})\\ &= \log\int_{\mathbf{W}} p(\mathbf{X},\mathbf{W})\cdot\frac{q(\mathbf{W})}{q(\mathbf{W})}\\ &= \log\left(\mathbb{E}_q\left[\frac{p(\mathbf{X},\mathbf{W})}{q(\mathbf{W})}\right]\right)\\ &\geq \mathbb{E}_q\left[\log p(\mathbf{X},\mathbf{W})\right] - \mathbb{E}_q\left[\log q(\mathbf{W})\right]. \quad \text{(Jensen's inequality)}\end{aligned} \tag{4.8}$$

Since the marginal $\log p(\mathbf{X})$ is fixed with respect to the variational distribution, minimizing the KL divergence in (4.7) is equivalent to maximizing the ELBO. In other words, tight lower bound of marginal evidence means that the KL divergence becomes small, and thus the true $p(\cdot)$ and the chosen $q(\cdot)$ are similar.

Mean Field Variational Inference (MFVI). MFVI is a kind of VI that assumes the variational family factorization, i.e., all latent variables are independent of each other. Specifically in our problem formulation, we assume that the variational joint distribution over the latent variables can be fully factorized as

$$q(\mathbf{Z},\mathbf{W}) = \prod_{n=1}^{N} q(\mathbf{z}_n|\lambda_{\mathbf{z}_n})q(\mathbf{W}|\lambda_{\mathbf{W}}), \tag{4.9}$$

where $q(\mathbf{W}|\lambda_{\mathbf{W}})$ and $q(\mathbf{z}_n|\lambda_{\mathbf{z}_n})$ are set to be in the same exponential family as the conditional distributions $p(\mathbf{W}|\mathbf{X},\mathbf{Z},\Omega_{\mathbf{W}})$ (4.2) and $p(\mathbf{Z}|\mathbf{X},\mathbf{W},\Omega_{\mathbf{Z}})$ (4.3). Here, $\lambda_{\mathbf{W}}$ and $\lambda_{\mathbf{Z}} = \{\lambda_{\mathbf{z}_n}, n = 1..N\}$ are the variational parameters. These parameters can be determined by maximizing the ELBO which is equivalent to minimizing the KL divergence, i.e., $\mathrm{KL}(q(\mathbf{Z},\mathbf{W})\|p(\mathbf{Z},\mathbf{W}|\mathbf{X},\Omega_{\mathbf{Z}},\Omega_{\mathbf{W}}))$. Specifically, one can decide the parameters by applying the coordinate ascent algorithm to the following objective, i.e., iteratively optimizing each variational distribution fixing the others:

$$\begin{aligned}\mathcal{L}(\lambda_{\mathbf{Z}}, \lambda_{\mathbf{W}}) &= \mathbb{E}_q\left[\log p(\mathbf{X}, \mathbf{Z}, \mathbf{W}|\Omega_{\mathbf{Z}}, \Omega_{\mathbf{W}})\right] - \mathbb{E}_q[\log q(\mathbf{Z}, \mathbf{W})] \\ &= \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n, \mathbf{W})}\left[\log p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{W})\right] \\ &\quad + \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n)}\left[\log p(\mathbf{z}_n|\Omega_{\mathbf{Z}})\right] + \mathbb{E}_{q(\mathbf{W})}\left[\log p(\mathbf{W}|\Omega_{\mathbf{W}})\right] \\ &\quad - \left\{\sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n)}[\log q(\mathbf{z}_n|\lambda_{\mathbf{z}_n})] + \mathbb{E}_{q(\mathbf{W})}[\log q(\mathbf{W}|\lambda_{\mathbf{W}})]\right\}. \end{aligned} \tag{4.10}$$

Decentralized Setting and Distributed MFVI (D-MFVI). For the decentralized version of the problem, we consider the similar configuration and notations as in the non-Bayesian setting from the previous chapter. The whole sensor network is represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $i, j \in \mathcal{V}$ denote vertices and $e_{ij} = (i, j) \in \mathcal{E}$ denotes the edge between the two vertices. Each i-th node has its own set of observations $\mathbf{X}_i = \{\mathbf{x}_{in}|n = 1..N_i\}$ and local latent variables $\mathbf{Z}_i = \{\mathbf{z}_{in}|n = 1..N_i\}$. In addition, each node maintains a local copy $\mathbf{W}_i$ of the global latent variable $\mathbf{W}$.

Each node approximates the posterior distributions over both global and local latent variables using the locally available information. Computing the posterior over local latent variables, $\mathbf{Z}_i$ is similar to the non-Bayesian case in the previous chapter, since they only depend on the corresponding local observations $\mathbf{X}_i$ and are independent from observations in the other nodes, given that the global latent variable $\mathbf{W}_i$ is fixed. On the other hand, finding the posterior over the local copies of the global latent variable $\mathbf{W}_i$ is a little bit different. In a non-Bayesian setting, the global latent variable was a parameter, thus the point estimates can be obtained by imposing consensus constraint, i.e., a series of equality constraints $\mathbf{W}_1 = \mathbf{W}_2 = \cdots = \mathbf{W}_{|\mathcal{V}|}$ on the global objective, as shown in (3.2). However, in Bayesian formulation, the parameters are random variables thus the notion of equality is replaced with one of the several notions of random variable equivalence, e.g., strict equality, equality in mean, almost sure equality, or equality in distribution. Here, we use the notion of equality in distribution, i.e., two random variables are equal if their cumulative distribution functions are equal.

Using the equality in distribution, the distributed MFVI (D-MFVI) can be formulated by imposing consensus constraints on the variational parameters of the posterior distributions over the local copies of the global variable as $\lambda_{\mathbf{W}_1} = \lambda_{\mathbf{W}_2} = \cdots = \lambda_{\mathbf{W}_{|\mathcal{V}|}}$ on the objective (4.10). Note the difference from the non-Bayesian case, where we are directly imposing consensus constraints on the local copies of the global parameter $\mathbf{W}_i$. The constrained optimization problem is to find the variational parameters by solving

$$[\hat{\lambda}_{\mathbf{Z}}, \hat{\lambda}_{\mathbf{W}}] = \underset{\lambda_{\mathbf{Z}_i}, \lambda_{\mathbf{W}_i} : i \in \mathcal{V}}{\arg\min} \quad - \left\{ \mathbb{E}_q \left[\log p(\mathbf{X}, \mathbf{Z}, \mathbf{W} | \Omega_{\mathbf{Z}}, \Omega_{\mathbf{W}}) \right] - \mathbb{E}_q \left[\log q(\mathbf{Z}, \mathbf{W}) \right] \right\},$$
$$s.t. \quad \lambda_{\mathbf{W}_i} = \rho_{ij}, \quad \rho_{ij} = \lambda_{\mathbf{W}_j}, \quad i \in \mathcal{V}, j \in \mathcal{B}_i, \tag{4.11}$$

where we introduced the auxiliary variables ρ_{ij} for each each e_{ij} to fully decentralize the computation, similar to the non-Bayesian approach. In the previous chapter, we employed ADMM to solve similar constrained optimization problems efficiently. As explained in the previous chapter and Appendix A, we derive augmented Lagrangian with a linear and a quadratic penalty terms of the objective, then apply coordinate descent with respect to each variable.

However, there are issues in generalizing this approach to the Bayesian formulation we are considering. First, even if we use the conjugate exponential family for prior and likelihood distribution choices, it is possible that the squared norm difference of $(\lambda_{\mathbf{W}_i} - \rho_{ij})$ in penalty terms may result in non-analytic updates for $\lambda_{\mathbf{W}_i}$, which lead to inefficient computation[1]. Second, it is unclear whether the simple squared Euclidean norm between variational parameters is enough to ensure the equality in distribution. We need a stronger justification or other evidence to ensure the equivalence.

To this end, we employee Bregman ADMM [6] (B-ADMM) rather than the standard ADMM for the optimization of the D-MFVI. Here, the global parameters are now the natural parameters of the exponential family distributions, and we propose to use the *log partition function* $\mathcal{A}_{\mathbf{W}}$ of the global parameter as the Bregman function, essentially replacing the quadratic norm difference term. These choices solve the two issues we discussed above. First, we can ensure that the base measure $\mathcal{A}_{\mathbf{W}}$ to be strictly convex, as

[1]Note that the updates for local latent variable $\lambda_{\mathbf{Z}_i}$ can be done in closed form since they do not need the equality constraints.

required by the definition of Bregman divergence, explained in Appendix C. It is worth noting that the log partition function $\mathcal{A}_{\mathbf{W}}$ is not a strictly convex function in general. However, we can make it strictly convex by ensuring that the exponential family is *minimal*[2] with reparameterization. With the strictly convex $\mathcal{A}_{\mathbf{W}}$, it can be shown that the coordinate updates for the variational parameters and Lagrange multipliers have an analytic solution. Second, by introducing the Bregman function, we can obtain additional theoretical support for our method. It is well known that the Bregman divergence between two parameters $\lambda_{\mathbf{W}_i}, \lambda_{\mathbf{W}_j}$ of the same minimal exponential family, say $p_{\mathcal{A}_{\mathbf{W}}}(\cdot)$, using the log partition function as the Bregman function, is equivalent to the reversed KL divergence between the exponential families [98, 99], i.e.,

$$B_{\mathcal{A}_{\mathbf{W}}}(\lambda_{\mathbf{W}_i}, \lambda_{\mathbf{W}_j}) = \mathrm{KL}(p_{\mathcal{A}_{\mathbf{W}}}(\mathbf{W}_j) || p_{\mathcal{A}_{\mathbf{W}}}(\mathbf{W}_i)) \approx \mathrm{KL}(q_{\mathcal{A}_{\mathbf{W}}}(\lambda_{\mathbf{W}_j}) || q_{\mathcal{A}_{\mathbf{W}}}(\lambda_{\mathbf{W}_i})). \quad (4.12)$$

If we assume that the auxiliary variable ρ_{ij} is the natural parameter of the same exponential family as $q(\mathbf{W}_i)$, then penalizing the difference between $\lambda_{\mathbf{W}_i}$ and ρ_{ij} using the Bregman divergence $B_{\mathcal{A}_{\mathbf{W}}}(\lambda_{\mathbf{W}_i}, \rho_{ij})$ can be an approximate way of minimizing the difference between the approximated posterior $q(\mathbf{W}_i|\lambda_{\mathbf{W}_i})$ and $q(\cdot|\rho_{ij})$ in a KL sense.

Therefore, we can now derive the analytic update formula solution for B-ADMM based D-MFVI problem (4.11) as follows:

$$\begin{aligned}
[\lambda_{\mathbf{Z}}^{(t+1)}, \lambda_{\mathbf{W}}^{(t+1)}] = \underset{\lambda_{\mathbf{Z}_i}, \lambda_{\mathbf{W}_i}: i \in \mathcal{V}}{\arg\min} \quad & -\sum_{i=1}^{|\mathcal{V}|} \sum_{n=1}^{N_i} \mathbb{E}_{q(\mathbf{z}_{in}, \mathbf{W}_i)} \left[\log p(\mathbf{x}_{in}|\mathbf{z}_{in}, \mathbf{W}_i)\right] \\
& -\sum_{i=1}^{|\mathcal{V}|} \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_{in})} \left[\log p(\mathbf{z}_{in}|\Omega_{\mathbf{z}})\right] \\
& -\frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \mathbb{E}_{q(\mathbf{W}_i)} \left[\log p(\mathbf{W}_i|\Omega_{\mathbf{W}})\right] \\
& +\sum_{i=1}^{|\mathcal{V}|} \sum_{n=1}^{N_i} \mathbb{E}_{q(\mathbf{z}_{in})}[\log q(\mathbf{z}_{in})] + \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \mathbb{E}_{q(\mathbf{W}_i)}[\log q(\mathbf{W}_i)] \\
& +\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{B}_i} \left(\gamma_{ij1}^{(t)\top}(\lambda_{\mathbf{W}_i} - \rho_{ij}^{(t)}) + \gamma_{ij2}^{(t)\top}(\rho_{ij}^{(t)} - \lambda_{\mathbf{W}_j})\right) \\
& +\eta \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{B}_i} B_{\mathcal{A}_{\mathbf{W}}}(\lambda_{\mathbf{W}_i}, \rho_{ij}^{(t)}) \qquad (4.13)
\end{aligned}$$

[2]An exponential family is minimal if the functions $\psi(.)$ and the statistics $\mathcal{T}(.)$ each are linearly independent.

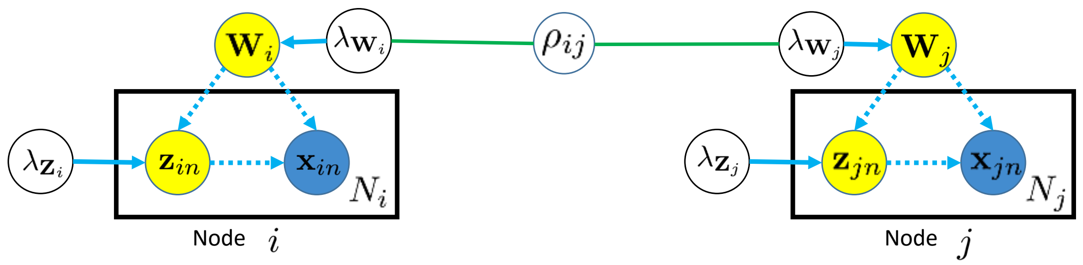


Figure 4.2: A graphical representation of the model of D-MFVI.

$$\rho^{(t+1)} = \arg\min_{\rho} \quad \eta \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{B}_i} B_{\mathcal{A}_{\mathbf{W}}}(\rho_{ij}, \lambda^{(t+1)}_{\mathbf{W}_i})$$

$$+ \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{B}_i} \left(\gamma^{(t)\top}_{ij1}(\lambda^{(t+1)}_{\mathbf{W}_i} - \rho_{ij}) + \gamma^{(t)\top}_{ij2}(\rho_{ij} - \lambda^{(t+1)}_{\mathbf{W}_j}) \right), \tag{4.14}$$

$$\gamma^{(t+1)}_{ijk} = \gamma^{(t)}_{ijk} + \eta(\lambda^{(t+1)}_{\mathbf{W}_i} - \rho^{(t+1)}_{ij}), \tag{4.15}$$

where $i \in V, j \in \mathcal{B}_i$ and γ_{ijk} with $k = 1, 2$ are the Lagrange multipliers. The overall iterative algorithm runs in coordinate descent fashion, similar to the non-Bayesian case. The learning rate parameter η should be predetermined [15]. The Bregman divergence induced by $\mathcal{A}_{\mathbf{W}}$ is denoted as $B_{\mathcal{A}_{\mathbf{W}}}(\cdot, \cdot)$ and formally, we define it as

$$B_{\mathcal{A}_{\mathbf{W}}}(x, y) = \mathcal{A}_{\mathbf{W}}(x) - \mathcal{A}_{\mathbf{W}}(y) - \langle x - y, \nabla \mathcal{A}_{\mathbf{W}}(y) \rangle, \tag{4.16}$$

where x, y can be replaced with the parameters of the exponential family distributions. Figure 4.2 shows an overlook of the D-MFVI formulation.

It should be noted that the Bregman divergence is not necessarily convex in the second argument. Thus, we cannot use $B_{\mathcal{A}_{\mathbf{W}}}(\lambda^{(t+1)}_{\mathbf{W}_i}, \rho_{ij})$ for the Bregman penalization term in Eq. 4.14. Instead, we use the reversed Bregman divergence ($B_{\mathcal{A}_{\mathbf{W}}}(\rho_{ij}, \lambda^{(t+1)}_{\mathbf{W}_i})$) as proposed by Wang and Banerjee[6]. They also proved the convergence of the new update formula. A brief review on Bregman divergence and Bregman ADMM is provided in Appendix C.

4.1.2 Example: Distributed Bayesian PCA

In this section, we derive a distributed counterpart of Bayesian PCA [88] as an example of the proposed D-MFVI.

Bayesian PCA (BPCA). We start from the Probabilistic PCA (PPCA) from the previous chapter, using the same notataion. In PPCA, we assume a local latent variable $\mathbf{z}_n \sim \mathcal{N}(\mathbf{z}_n|0, \mathbf{I})$, with a generative relation

$$\mathbf{x}_n = \mathbf{W}\mathbf{z}_n + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{4.17}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|0, a^{-1}\mathbf{I})$ and a is the noise precision. The likelihood and the posterior inference are, similarly to (3.9),

$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{W}\mathbf{z}_n + \boldsymbol{\mu}, a^{-1}\mathbf{I}), \tag{4.18}$$

$$p(\mathbf{z}_n|\mathbf{x}_n) = \mathcal{N}(\mathbf{z}_n|\mathbf{L}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x}_n - \boldsymbol{\mu}), a^{-1}\mathbf{L}^{-1}), \tag{4.19}$$

where $\mathbf{L} = \mathbf{W}^{\mathrm{T}}\mathbf{W} + a^{-1}\mathbf{I}$. Then we find the set of optimal parameters $\mathbf{W}, \boldsymbol{\mu}, a$ by searching the maximum likelihood estimates of the marginal data likelihood, or by applying the EM algorithm on expected complete data log likelihood with respect to the posterior density $p(\mathbf{Z}|\mathbf{X})$. In the Bayesian formulation of PCA, we first introduce a prior distribution $p(\mathbf{W}, \boldsymbol{\mu}, a)$ over the parameters of the model. Then, we compute the corresponding posterior distribution $p(\mathbf{W}, \boldsymbol{\mu}, a|\mathbf{X})$ by multiplying the prior by the likelihood function Eq. 4.18 and normalizing it.

To do the BPCA inference, we need to make two important decisions: (a) the choice of the prior distribution, and (b) a tractable algorithm for computing the posterior distribution. In BPCA, we typically assume that the parameters are independent from each other as

$$p(\mathbf{W}, \boldsymbol{\mu}, a) = p(\mathbf{W})p(\boldsymbol{\mu})p(a) \tag{4.20}$$

and we further assume that the noise precision a is a fixed but unknown parameter for simplicity. We define an independent Gaussian prior over each row of the global parameter $\mathbf{W}$ as

$$p(\mathbf{W}|\alpha) = \prod_{d=1}^{D} \left(\frac{\alpha_d}{2\pi}\right)^{M/2} \exp\left\{-\frac{\alpha_d}{2} \cdot \left(\mathbf{w}_d^\top - \bar{\mathbf{w}}_d^\top\right) \cdot \left(\mathbf{w}_d^\top - \bar{\mathbf{w}}_d^\top\right)^\top\right\}, \tag{4.21}$$

where $\mathbf{w}_d^\top$ is the d-th row of $\mathbf{W}$. $\bar{\mathbf{w}}_d$ and $\alpha_d, d = 1..D$ are the mean and the precision hyperparameters, respectively. We consider another Gaussian prior for $\boldsymbol{\mu}$ as

$$p(\boldsymbol{\mu}) = \mathcal{N}(\bar{\boldsymbol{\mu}}, \tau^{-1}\mathbf{I}), \tag{4.22}$$

where $\bar{\boldsymbol{\mu}}$ and τ are the mean and the precision hyperparameters, respectively. Since computing the exact posterior distribution is intractable, we use MFVI. We assume a fully factorizable posterior of the form

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\mu}) = \prod_{d=1}^{D} \prod_{m=1}^{M} q(\mathbf{w}_{dm}) \prod_{n=1}^{N} q(\mathbf{z}_n) \prod_{d=1}^{D} q(\boldsymbol{\mu}_d). \tag{4.23}$$

Since we employed conjugate priors for $\mathbf{W}, \mathbf{Z}$ and $\boldsymbol{\mu}$, the posterior distributions are also Gaussian. We denote the mean and precision of the posterior distributions as

$$q(\mathbf{w}_{dm}) \sim \mathcal{N}(m^{\mathbf{w}}_{dm}, \lambda^{\mathbf{w}}_{dm}) \tag{4.24}$$

$$q(\mathbf{z}_n) \sim \mathcal{N}(m^{\mathbf{z}}_{n}, \lambda^{\mathbf{z}}_{n})) \tag{4.25}$$

$$q(\boldsymbol{\mu}_d) \sim \mathcal{N}(m^{\boldsymbol{\mu}}_{d}, \lambda^{\boldsymbol{\mu}}_{dm}). \tag{4.26}$$

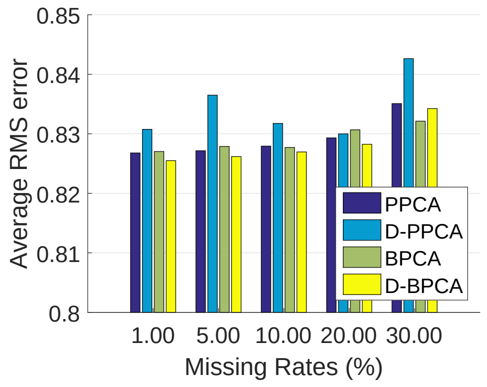
Distributed BPCA (D-BPCA). It is straightforward to apply D-MFVI to the BPCA model. We have two global latent variables $\mathbf{W}, \boldsymbol{\mu}$ and a local latent variable $\mathbf{z}_n$. As in D-PPCA, we distribute samples across the nodes in the network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the inference is done by each node and its local communications between one-hop neighbors. We define the local set of model parameters as

$$\Xi_i = \left\{ (m^{\mathbf{w}}_{dm})_i \,, (\lambda^{\mathbf{w}}_{dm})_i \,, \left(m^{\boldsymbol{\mu}}_{d}\right)_i \,, \left(\lambda^{\boldsymbol{\mu}}_{dm}\right)_i \,, (m^{\mathbf{z}}_{n})_i \,, (\lambda^{\mathbf{z}}_{n})_i \right\}. \tag{4.27}$$

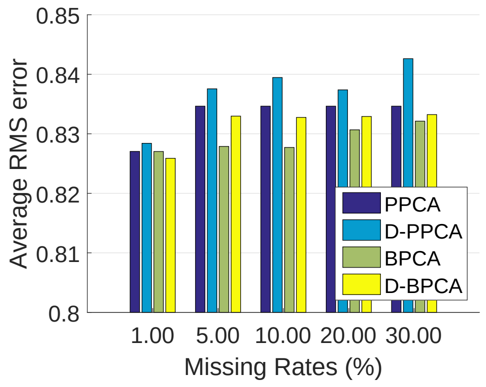
Then, the D-MFVI optimization problem can be written as

$$\begin{aligned}
\hat{\Xi} = \underset{\Xi_i : i \in \mathcal{V}}{\arg\min} \quad & -\sum_{i=1}^{|\mathcal{V}|} \left\{ \mathbb{E}_q\big[\log p(\mathbf{X}_i, \mathbf{Z}_i, \mathbf{W}_i | a, \alpha, \tau, \bar{\boldsymbol{\mu}}, \bar{\mathbf{w}}_d)\big] - \mathbb{E}_q\left[\log q(\mathbf{W}_i, \boldsymbol{\mu}_i, \mathbf{Z}_i)\right] \right\} \\
s.t. \quad & (m^{\mathbf{w}}_{dm})_i = (\rho^{\mathbf{w}}_{dm})_{ij}, \quad (\rho^{\mathbf{w}}_{dm})_{ij} = (m^{\mathbf{w}}_{dm})_j, \\
& (\lambda^{\mathbf{w}}_{dm})_i = (\gamma^{\mathbf{w}}_{dm})_{ij}, \quad (\gamma^{\mathbf{w}}_{dm})_{ij} = (\lambda^{\mathbf{w}}_{dm})_j \\
& (m^{\boldsymbol{\mu}}_{d})_i = (\rho^{\boldsymbol{\mu}}_{d})_{ij}, \quad (\rho^{\boldsymbol{\mu}}_{d})_{ij} = (m^{\boldsymbol{\mu}}_{d})_j, \\
& (\lambda^{\boldsymbol{\mu}}_{d})_i = (\gamma^{\boldsymbol{\mu}}_{d})_{ij}, \quad (\gamma^{\boldsymbol{\mu}}_{d})_{ij} = (\lambda^{\boldsymbol{\mu}}_{d})_j,
\end{aligned}$$

where $i \in \mathcal{V}, j \in \mathcal{B}_i, d = 1..D$ and $\{(\rho^{\mathbf{w}}_{dm})_{ij}, (\gamma^{\mathbf{w}}_{dm})_{ij}, (\rho^{\boldsymbol{\mu}}_{d})_{ij}, (\gamma^{\boldsymbol{\mu}}_{d})_{ij}\}$ are auxiliary variables of edge (i, j). Note that we plug in the same hyperparameters to all nodes, and this can be determined in advance. The full derivation of the coordinate descent update formula for solving the above optimization problem can be found in the extended version



(a) Missing At Random



(b) Missing Not At Random

Figure 4.3: Average root mean squared error of reconstructions based on PPCA, D-PPCA, BPCA, and D-BPCA results.

of the prior publication [3]. Generalizing our distributed BPCA (D-BPCA) to deal with missing data is straightforward and follows the prior work [84], as we described in the previous chapter.

The empirical convergence property of D-BPCA is similar to that of D-PPCA, robust to number of nodes, η value choice, and the graph topology [3]. We provide experimental results on synthetic data with missing values here. We generated 500 samples of 20-dimensional input data to find 5-dimensional subspaces. As before, we conducted experiments using 20 different initializations and took the average. Figure 4.3 shows the comparison of PPCA, D-PPCA, BPCA, and D-BPCA methods. One can see that both centralized and distributed Bayesian methods outperform non-Bayesian counterparts. Moreover, the difference between centralized and distributed methods becomes smaller in the Bayesian case, which demonstrates the additional benefit of the Bayesian approach in the distributed setting.

4.1.3 Application: Distributed Computer Vision

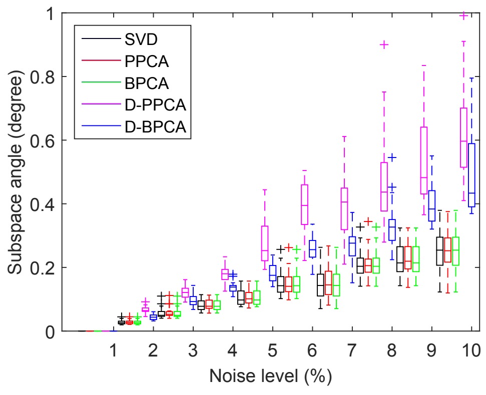
We apply our model to a set of Structure from Motion (SfM) problems. We compared our distributed algorithm with traditional SVD, Centralized PCA (PPCA) [11], Distributed PPCA (D-PPCA) [1], and Centralized BPCA (BPCA) [84]. We used the same experimental setup as in the D-PPCA case from the previous chapter.

The estimates of $\mathbf{W}$ and $\mathbf{Z}$ can be found using the D-BPCA, where $\mathbf{W}$ is treated as the latent, common 3D structure, and $\mathbf{Z}$ is the latent local camera motion. It is worth noting that D-PPCA can only provide the uncertainty around the motion matrix $\mathbf{Z}$, while D-BPCA can obtain additional estimates of the variance of the 3D structure $\mathbf{W}$ as a consensus among cameras. We show that our D-BPCA can be used as an effective framework for the distributed affine SfM problem. For all SfM experiments, the network has five nodes connected with a ring topology, and $\eta = 10$.

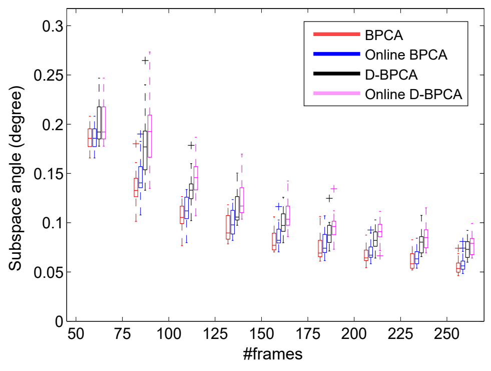
Following [60], we equally partitioned the frames into the five nodes to simulate five cameras. The algorithm stops when there is no more than 10^{-3} relative change in the objective of (4.11). For the quantitative measure, we computed the maximum subspace angle between the ground truth 3D coordinates (in case of synthetic data experiments) and the estimated 3D structure matrix $\mathbf{W}$ to compare performance of the algorithms. For the BPCA and D-BPCA, we used the posterior mean of the 3D structure matrix to calculate this subspace angle.

Results on Synthetic Data Similar to the synthetic experiments conducted in the previous chapter, we used the environment containing one unit cube rotating around the origin for the target object, with 5 cameras facing the cube in a 3D space. However, unlike the setting for D-PPCA, we rotated the cube every $3°$ over $150°$ clockwise in order to obtain additional views necessary for our online learning evaluation. Therefore, each camera observed 50 frames in this setting. Figure 4.4a shows the performance comparison of different models in the case of noisy data. We ran 20 independent runs with different initializations, each with 10 different noise levels. As shown in the figure, D-BPCA consistently outperforms D-PPCA for noisy input data, thanks in part to improved robustness to overfitting.

For the MAR experiment, we randomly removed 20% of data points over 10 independent trials. The average subspace differences between D-PPCA versus the ground truth and D-BPCA versus the ground truth were $1.41°$ and $1.01°$, respectively. The same set of experiments was done for the MNAR case, with the missing data generated by the realistic visual occlusion. The subspace angle differences were $17.66°$ for D-PPCA and $14.12°$ for D-BPCA, respectively. Again, D-BPCA performed better than



(a) Noisy data experiment



(b) Online data experiment

Figure 4.4: Results for the cube synthetic data (crosses denote outliers).

D-PPCA, although the error rates were higher in the more challenging MNAR case.

One particular advantage of the D-BPCA over D-PPCA and the SVD is its ability to naturally conduct online Bayesian estimation in the distributed setting. We first used 10 frames in each camera as the first batch of the data. Then, we added 5 more frames to each camera, repeatedly. Results over 10 different trials with 1% input noise are shown in Figure 4.4b. PPCA models are omitted because their non-Bayesian nature obstructs an easy application to the online setting. Figure 4.4b demonstrates that the subspace angle error of the online D-BPCA closely follows centralized BPCA in accuracy.

Results on Real Data We demonstrate the usefulness of the proposed D-BPCA for real data using the Caltech 3D Objects on Turntable dataset [64] and Hopkins155 dataset [66]. Following [1], we selected 5 objects from the Caltech dataset. For the Hopkins155 dataset, we selected 90 single-object sequences. For both datasets, we used the same setup as described in [1]. The subspace angles between the structure estimated using the centralized SVD-based SfM, and the distributed algorithms for the Caltech dataset, are summarized in Table 4.1. For this case, we ran 20 independent initializations to obtain the mean and the variance. 10% MAR and MNAR results are also provided in the left two columns of Table 4.2. They are averaged over all 5 objects we tried. As one can see, the D-BPCA's performance was consistently better than the D-PPCA's. The right two columns of Table 4.2 shows the results for the Hopkins dataset. Numbers are averages of maximum subspace angles between D-PPCA and

Table 4.1: Results of Caltech dataset. All results ran 20 independent initializations.

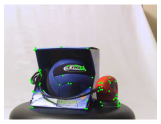
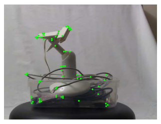
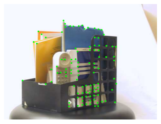
Object	BallSander	BoxStuff	Rooster	Standing	StorageBin
					
# Points	62	67	189	310	102
# Frames	30	30	30	30	30
Subspace angle between centralized SVD SfM and D-PPCA (degree)					
Mean	1.4934	1.4402	1.4698	2.6305	0.4513
Variance	0.4171	0.4499	0.9511	1.7241	1.2101
Subspace angle between centralized SVD SfM and D-BPCA (degree)					
Mean	**0.9910**	**0.9879**	**1.3855**	**0.9621**	**0.4203**
Variance	0.0046	0.0986	0.0080	0.0033	0.0044

Table 4.2: Results of Caltech and Hopkins datasets with missing values.

		Caltech		Hopkins	
		MAR	MNAR	No-missing	MAR
D-PPCA	Mean	4.0609	9.4920	3.9523	13.4753
	Variance	1.2976	5.9624	3.3119	12.9832
D-BPCA	Mean	**2.2012**	**7.2187**	**0.7975**	**6.4372**
	Variance	1.3179	5.2853	0.5684	5.0689

D-BPCA versus SVD-based SfM, for all 90 objects. We also provide results with 10% MAR. We did not conduct MNAR experiments on Hopkins because the ground truth occlusion information is not provided with the dataset. We ran 5 independent trials. Again, D-BPCA consistently performs better than D-PPCA. It should be noted that although the subspace angle difference is large for the MAR case for D-PPCA and D-BPCA, estimated 3D structures were similar to that of SVD up to the orthogonal ambiguity.

4.2 Faster Optimization for Distributed Learning

So far, we have considered the probabilistic modeling aspect of distributed learning. These models depend on EM-like algorithms to optimize, and these are iterative by nature. Thus, it is desirable to have the algorithm converge as quickly as possible. To this end, we investigate another aspect of our distributed learning framework in this section: improvement of the distributed optimization convergence speed.

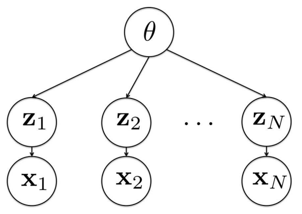
4.2.1 Need of Faster Convergence for Distributed Learning

The need for algorithms and methods that can handle large data in a distributed setting has grown significantly in recent years. For both induced and instrically distributed data scenarios we explained in Chapter 1, several parallel and distributed learning approaches have been proposed to meet these needs, as introduced in Chapter 2. In particular, the ADMM is an optimization technique that has been widely used in computer vision and machine learning to handle model estimation and learning, in either of the two large data settings [100, 101, 102, 103, 104, 105, 106, 107].

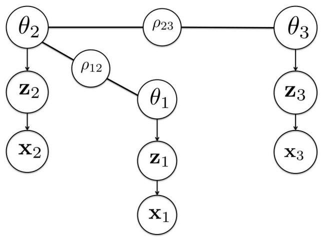
In the distributed optimization setting, the distributed nodes process data locally by solving small optimization problems and aggregate the result by exchanging the (possibly compressed) local solutions (e.g., local model parameter estimates) to arrive at a consensus global result. However, the nature of distributed learning models, particularly in the fully distributed setting where no network topology is presumed, inherently requires repetitive communications between the device nodes. Therefore, it is desirable to reduce the amount of information exchanged and simultaneously improve computational efficiency through faster convergence of such distributed algorithms. Our methods can be applied to any arbitrary distributed settings as well as parallel computation that requires a certain centralized connection (e.g., a star topology).

To this end, the contributions of this section are threefolds:

- We propose variants of ADMM for the consensus-based distributed learning that are faster than the standard ADMM. Our method extends an acceleration approach for ADMM [108] by an efficient variable penalty parameter update strategy. This strategy results in improved convergence properties of ADMM, and also works in a fully distributed fashion.



(a) Centralized



(b) Distributed

Figure 4.5: Centralized and decentralized models for probabilistic PCA.

- We extend our proposed method to automatically determine the maximum number of iterations allocated to successive updates by employing a budget management scheme. This strategy results in adaptive parameter tuning for ADMM, removing the need for arbitrary parameter settings, and effectively induces a varying network communication topology.

- We apply the proposed method to a prototypical vision and learning problem, the distributed PPCA for structure-from-motion, and demonstrate its empirical utility over the traditional ADMM.

4.2.2 Improving Empirical Convergence of ADMM Optimization

Recall the consensus-based distributed optimization problem we introduced in Chapter 2.1. As explained in the previous chapter, the optimization can be approached efficiently by exploiting the ADMM. Restating it, the optimization problem can be written as

$$\arg\min_{\theta_i} \quad \sum_{i=1}^{J} f_i(\theta_i) \quad s.t. \quad \theta_i = \theta_j, \quad \forall i \neq j. \tag{4.28}$$

The above consensus formulation is particularly suitable for many optimization problems that appear in computer vision. For instance, since $f_i(\theta_i)$ can be any convex function, we can also consider a probabilistic model estimation problem with the joint negative log likelihood $f_i(\theta_i) = -\log p(x_i, z_i|\theta_i)$ between the observation x_i and the corresponding latent variable z_i. Assuming (x_i, z_i) are independent and identically distributed, finding the maximum likelihood estimate of the shared parameter $\bar{\theta}$ can then be formulated as the optimization problem we described above for many exponential family parametric densities. Moreover, the function need not be a likelihood, but can also be a typical decomposable and regularized loss that occurs in many vision problems such as denoising or dictionary learning.

It is often very convenient to consider the above consensus optimization problem from the perspective of optimization on graphs. For instance, the centralized i.i.d. Maximum Likelihood learning can be viewed as the optimization on the graph in Figure 4.5a. Edges in this graph depict functional (in)dependencies among variables, commonly found in representations such as Markov Random Fields [107] or Factor Graphs [109]. In this context, to fully decompose $f(\cdot)$ and eliminate the need for a processing center completely, one can introduce auxiliary variables ρ_{ij} on every edge to break the dependency between θ_i and θ_j [36, 1], as shown in Figure 4.5. This generalizes to arbitrary graphs, where the connectivity structure may be implied by node placement or communication constraints (camera networks), imaging constraints (pixel neighborhoods in images or frames in a video sequence), or other contextual constraints (loss and regularization structure).

In general, given a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the nodes $i, j \in \mathcal{V}$ and the edges $e_{ij} = (i, j) \in \mathcal{E}$, the consensus optimization problem becomes

$$\min \quad \sum_{i \in \mathcal{V}} f_i(\theta_i) \quad s.t. \quad \theta_i = \rho_{ij}, \quad \rho_{ij} = \theta_j, j \in \mathcal{B}_i. \tag{4.29}$$

Solving that problem is equivalent to optimizing the augmented Lagrangian

$$\begin{aligned}\mathcal{L}(\mathbf{\Theta}) = \sum_{i \in \mathcal{V}} \mathcal{L}_i(\mathbf{\Theta}_i) = f_i(\theta_i) + \sum_{j \in \mathcal{B}_i} \left\{ \lambda_{ij1}^{\top}(\theta_i - \rho_{ij}) + \lambda_{ij2}^{\top}(\rho_{ij} - \theta_j) \right\} \\ + \frac{\eta}{2} \sum_{j \in \mathcal{B}_i} \left\{ \|\theta_i - \rho_{ij}\|^2 + \|\rho_{ij} - \theta_j\|^2 \right\},\end{aligned} \tag{4.30}$$

where $\mathbf{\Theta} = \{\mathbf{\Theta}_i : i \in \mathcal{V}\}$, $\mathbf{\Theta}_i = \{\theta_i, \rho_i, \lambda_i\}$ are parameters to find, $\lambda_i = \{\lambda_{ij1}, \lambda_{ij2} : j \in \mathcal{B}_i\}$, λ_{ij1}, λ_{ij2} are Lagrange multipliers, $\mathcal{B}_i = \{j | e_{ij} \in \mathcal{E}\}$ is the set of one-hop neighbors of node i, $\eta > 0$ is a fixed scalar penalty constraint, and $\| \cdot \|$ is induced norm. The ADMM approach suggests that the optimization can be done in a coordinate descent fashion, taking the gradient of each variable while fixing all the others.

Residual Balancing (RB) : The currently known convergence rate of ADMM is $O(1/T)$, where T is the number of iterations [110]. Even though $O(1/T)$ is the best known bound, it has been observed empirically that ADMM converges more quickly in many applications. Moreover, the computation time per each iteration may dominate the total algorithm running time. Thus many speed-up techniques for ADMM have been proposed that are application specific. One way is to develop a predictor-corrector step for the coordinate descent [111], using some available acceleration method, such as the accelerated gradient method [112]. This guarantees quadratic convergence for strongly convex $f_i(\cdot)$. Another way is to replace the gradient descent optimization with a stochastic one [113, 114]. This approach has recently gained attention as it greatly reduces the computation per iteration. However, these methods usually require the coordinating center node, and thus may not be readily applicable to the decentralized setting. Moreover, we want to preserve the application range of ADMM and avoid introducing additional assumptions on $f_i(\cdot)$.

One way to improve convergence speed of ADMM is through the use of different constraint penalty in each iteration. For example, a variant of ADMM with self-adaptive penalty [108], called Residual Balancing (RB) [115], improved the convergence speed as well as made its performance less dependent on initial penalty values. The idea is to change the constraint penalty, taking account of the relative magnitudes of *primal* and *dual* residuals of ADMM, as follows:

$$\eta^{t+1} = \begin{cases} \eta^t \cdot (1+\tau^t) & \text{, if } \|r^t\|_2 > \mu\|s^t\|_2 \\ \eta^t \cdot (1+\tau^t)^{-1} & \text{, if } \|s^t\|_2 > \mu\|r^t\|_2 \\ \eta^t & \text{, otherwise} \end{cases} \tag{4.31}$$

where t is the iteration index, $\mu > 1$, $\tau^t > 0$ are parameters, r^t and s^t are the primal and dual residuals, respectively. The definitions of these residuals can be found in Appendix A. The primal residual measures the violation of the consensus constraints and the dual residual measures the progress of the optimization in the dual space. This update converges when τ^t satisfies $\sum_{t=0}^{\infty} \tau^t < \infty$, i.e., we stop updating η^t after a finite number of iterations. A typical choice for parameters are suggested as $\mu = 10$ and

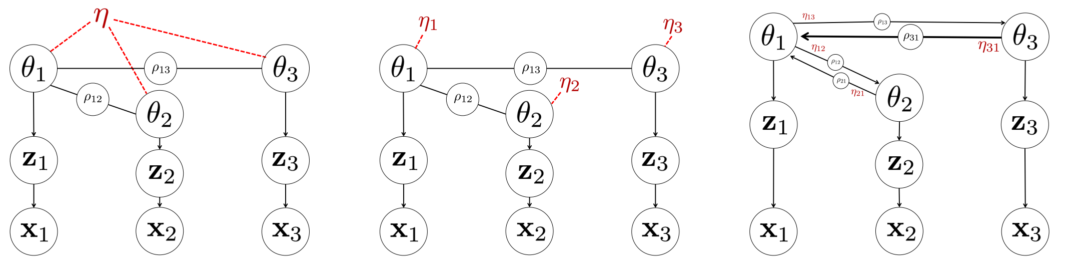


Figure 4.6: Centralized, distributed, and proposed learning models in a chain network. The bigger size of ρ_{ij} means that the corresponding constraint is more penalized.

$\tau^t = 1$ at all t iterations.

The strength of this approach is that conservative changes in the penalty are guaranteed to converge [116, 15]. However, similar to other ADMM speed-up approaches mentioned above, this update scheme relies on the global computation of the primal and the dual residuals, and requires the η^t stored in nodes to be homogeneous over the entire network. Thus, it is not a fully decentralized scheme, as depicted in Figure 4.6a. Moreover, the choice of parameters, as well as the maximum number of iterations, require manual tuning.

ADMM with Varying Penalty (ADMM-VP) : Throughout the chapter, the superscript t in all terms with subscript i denote either the objective function or parameter at the t-th iteration for node i. In order to extend the update strategy of (4.31) for a fully distributed setting, we first introduce η_i^t, the penalty for the i-th node at the t-th iteration. Next, we need to compute local primal and dual residuals for each node i. In the fully decentralized learning framework of [36, 1], the dual auxiliary variable vanishes from derivation. However, to compute the residuals, we need to keep track of the dual variable, which is essentially the average of local estimates, explicitly over iterations. The squared residual norms for the i-th node are defined as

$$\|r_i^t\|_2^2 = \|\theta_i^t - \bar{\theta}_i^t\|_2^2, \quad \|s_i^t\|_2^2 = (\eta_i^t)^2\|\bar{\theta}_i^t - \bar{\theta}_i^{t-1}\|_2^2, \tag{4.32}$$

where

$$\bar{\theta}_i^t = \frac{1}{|\mathcal{B}_i|}\sum_{j\in\mathcal{B}_i}\theta_j^t. \tag{4.33}$$

Note the difference from the standard residual definitions for consensus ADMM [15], used in (4.31), where the dual variable is considered as a single, globally accessible variable, $\bar{\theta}^t$ instead of local $\bar{\theta}_i^t$. This allows each node to change its η_i^t based on its own local residuals. The penalty update scheme is similar to (4.31) but η^t, $\|r^t\|_2$ and $\|s^t\|_2$ are replaced with η_i^t, $\|r_i^t\|_2$ and $\|s_i^t\|_2$, respectively. Lastly, the original adaptive penalty ADMM [108] stopped changing η^t after $t > 50$. However, in ADMM-VP, if we stop the same way, we end up with heterogeneously fixed penalty values, which impact the convergence of ADMM by yielding heavy oscillations near the saddle point. Therefore, we reset all penalty values in all nodes to a pre-defined value (e.g., η^0, the initial penalty parameter) after a fixed number of iterations. As we fix the penalty values homogeneously after a finite number of iterations, it becomes the standard ADMM after that point, thus the convergence of ADMM-VP update is guaranteed. Figure 4.6b depicts ADMM-VP and also contrasts with the standard RB method in Figure 4.6a.

ADMM with Network Adaptive Penalty (ADMM-NAP) : We propose further extensions of the ADMM-VP. We replace η_i by introducing a bi-directional graph with a penalty constraint parameter η_{ij} specific to directed edge e_{ij} from node i to j, as depicted in Figure 4.6c. The modified augmented Lagrangian $\mathcal{L}_i$ is similar to (4.30), except that we replace η with η_{ij}. The penalty constraint controls the amount each constraint contributes to the local minimization problem. The penalty constraint parameter η_{ij} is determined by evaluating the parameter θ_j from node j with the local objective function $f_i(\cdot)$ of node i as

$$\eta_{ij}^{t+1} = \begin{cases} \eta^0 \cdot (1 + \tau_{ij}^t) & \text{, if } t < t^{max} \\ \eta^0 & \text{, otherwise} \end{cases} \tag{4.34}$$

where t^{max} is the maximum number of iterations for the update as proposed in [108],

$$\tau_{ij}^t = \frac{\kappa_i^t(\theta_i^t)}{\kappa_i^t(\theta_j^t)} - 1\,, \tag{4.35}$$

$$\kappa_i^t(\theta) = \left(\frac{f_i^t(\theta) - f_i^{min}}{f_i^{max} - f_i^{min}} + 1 \right)\,, \tag{4.36}$$

$$f_i^{max} = \max\{f_i^t(\theta_i^t), f_i^t(\theta_j^t) : j \in \mathcal{B}_i\}\,, \tag{4.37}$$

$$f_i^{min} = \min\{f_i^t(\theta_i^t), f_i^t(\theta_j^t) : j \in \mathcal{B}_i\}\,. \tag{4.38}$$

In each iteration t, each i-th node will evaluate its local objective using its own estimate of θ_i^t, and the estimates from other neighboring nodes θ_j^t. Here, we use ρ_{ij}^t instead of actual θ_j^t to retain locality of each node from the neighbors. Then, we assign more weight to the neighbor with a better parameter estimate for the local $f_i(\cdot)$ (i.e., larger penalty η_{ij}^t if $f_i(\theta_j) < f_i(\theta_i)$) with the above update scheme. We refer to this extension as the ADMM-AP (adaptive penalty), and the intuition behind the ADMM-AP update is to emphasize the local optimization during early stages and then deal with the consensus update at later, subsequent stages. If all local parameters yield similarly valued local objectives $f_i(\cdot)$, the onus is placed on consensus. This makes ADMM-AP different from pre-initialization, which performs the local optimization using the local observations and ignores the consensus constraints.

Note that unlike the update strategy of (4.31), we do not need to specify τ^t manually and the weight is automatically chosen according to the normalized difference in the local objective evaluation among neighboring parameters and the local parameter at node i. The proposed algorithm also emphasizes the objective minimization over the minimization that solely depends on the norms of primal and dual residuals of constraints. The hope is that we not only achieve the consensus of the parameters of the model but also a *good* estimate with respect to the objective.

On the other hand, the convergence property of [108] still holds for the proposed algorithm. Following Remark 4.2 of [108], the requirement for the convergence is to satisfy the update ratio to be fixed after some $t^{\max} < \infty$ iterations. Moreover, the proposed update ensures bounding by $\eta_{ij}^{t+1}/\eta_{ij}^t \in [0.5, 2]$, which matches with the increase and decrease amount suggested in [108, 15]. One may use $t^{\max} = 50$, as in [108].

We propose a further extension of ADMM-AP so that it can automatically decide the maximum number of penalty updates, i.e., the number of ADMM iterations with penalty updates. We change the algorithm (4.34) into

$$\eta_{ij}^{t+1} = \begin{cases} \eta^0 \cdot (1 + \tau_{ij}^t) & , \text{if } \sum_{u=1}^{t} |\tau_{ij}^u| < \mathcal{T}_{ij}^t \\ \eta^0 & , \text{otherwise.} \end{cases} \tag{4.39}$$

In addition to the adaptive penalty update, the inequality condition on the summation of $\tau_{ij}^u, u = 1..t$ encodes the spent *budget* when the edge e_{ij} changed η_{ij}. All nodes have their budget's upper bound $\mathcal{T}_{ij}^t$, and everytime a node makes a change to η_{ij}, it has to *pay* exactly the amount they changed. If the edge has changed η_{ij} too much, or too often, the update strategy will block the edge from changing η_{ij} any more.

The update scheme is guaranteed to convergence if $\mathcal{T}_{ij}^t$ is simply set to constant $\mathcal{T}$ for all i, j, t or if $\tau_{ij}^t = 0$ for $t > t^{max}$. However, with a different objective function and varying network connectivity, a different upper bound should be required. This is because a given upper bound $\mathcal{T}$ or maximum iteration t^{max} could be too small for a certain node to fully take advantage of our adaptation strategy, or they could be too big so that it converges much more slowly due to the continuously changing η_{ij}^t. To this end, we propose an updating strategy for $\mathcal{T}_{ij}^t$ as

$$\mathcal{T}_{ij}^{t+1} = \begin{cases} \mathcal{T}_{ij}^t + \alpha^n \mathcal{T} & \text{, if } \sum_{u=1}^t |\tau_{ij}^u| \geq \mathcal{T}_{ij}^t \text{ and } \left| f_i(\theta_i^t) - f_i(\theta_i^{t-1}) \right| > \beta \\ \\ \mathcal{T}_{ij}^t & \text{, otherwise} \end{cases} \tag{4.40}$$

where $\mathcal{T}_{ij}^0$ is set by an initial parameter $\mathcal{T}$ and $\alpha, \beta \in (0, 1)$ are parameters. Whenever $\mathcal{T}_{ij}^{t+1} > \mathcal{T}_{ij}^t$, we increase n by 1. Once

$$\sum_{u=1}^t |\tau_{ij}^u| \geq \mathcal{T}_{ij}^t \tag{4.41}$$

but its objective value is still significantly changing, i.e., $\left| f_i(\theta_i^t) - f_i(\theta_i^{t-1}) \right| > \beta$, $\mathcal{T}_{ij}^{t+1}$ is increased by $\alpha^n \mathcal{T}$. Note that the independent upper bound $\mathcal{T}_{ij}^t$ for each η_{ij}^t update on the edge e_{ij} makes it sensitive to the various network topologies, but it still satisfies the convergence condition because

$$\lim_{t \to \infty} \mathcal{T}_{ij}^t \leq \sum_{n=1}^{\infty} \alpha^{n-1} \, \mathcal{T} = \frac{1}{1 - \alpha} \mathcal{T}. \tag{4.42}$$

We refer to this penalty update scheme as ADMM-NAP (network adaptive penalty). One can see that ADMM-AP is a special case of ADMM-NAP.

Combining and Choosing Criteria to Use : Observing (4.31) and the proposed update schemes (4.34) and (4.39), one can easily develop a combined update strategy by replacing τ^t in (4.31) with τ_{ij}^t. Based on preliminary experiments, we found that this replacement yields little utility. Instead, we suggest another penalty update strategy combining ADMM-VP and ADMM-AP as

$$\eta_{ij}^{t+1} = \begin{cases} \eta_{ij}^t \cdot (1+\tau_{ij}^t) \cdot 2 & \text{, if } \|r_i^t\|_2 > \mu \|s_i^t\|_2 \\ \eta_{ij}^t \cdot (1+\tau_{ij}^t) \cdot (1/2) & \text{, if } \|s_i^t\|_2 > \mu \|r_i^t\|_2 \\ \eta_{ij}^t & \text{, otherwise} \end{cases} \tag{4.43}$$

which we denote as ADMM-VP + AP. We reset $\eta_{ij}^t = \eta^0$ when $t > t^{\max}$. In order to combine ADMM-VP and ADMM-NAP, we consider the summation condition of τ_{ij}^t, as in (4.39). We denote this strategy as ADMM-VP + NAP.

The key difference between ADMM-AP and ADMM-NAP is that the latter does not require $t_{\max}$ to be decided in advance. If the best $t_{\max}$ is known for a certain application, there is no significant benefit of ADMM-NAP over ADMM-AP. However, in many real-world problems, $t_{\max}$ is not known, and ADMM-NAP can be an effective option, as demonstrated in Figure 4.8c.

4.2.3 Example: D-PPCA using RB-based Penalty Update Criteria

In this section, we show how our method can be applied to an existing distributed learning framework in the context of distributed probabilistic principal component analysis (D-PPCA). D-PPCA can be viewed as a fundamental approach to a general matrix factorization task in the presence of potentially missing data, with many applications in machine learning.

As explained in Chapter 3, the distributed extension of PPCA (D-PPCA) can be derived by applying ADMM to the centralized PPCA model above. Each node learns its local copy of PPCA parameters with its set of local observations $\mathbf{X}_i = \{\mathbf{x}_{in} | n = 1..N_i\}$, where $\mathbf{x}_{in}$ denotes the n-th observation in the i-th node, and N_i is the number of observations available in the node. Then, they exchange the parameters using the Lagrange multipliers and impose consensus constraints on the parameters. The global constrained optimization is

$$\min_{\mathbf{\Theta}_i} \quad -\log p(\mathbf{X}_i|\mathbf{\Theta}_i), \quad s.t. \quad \mathbf{\Theta}_i = \rho_{ij}^{\mathbf{\Theta}}, \rho_{ij}^{\mathbf{\Theta}} = \mathbf{\Theta}_j, \tag{4.44}$$

where $\mathbf{\Theta}_i = \{\mathbf{W}_i, \boldsymbol{\mu}_i, a_i\}$ is the set of local parameters and $\rho_{ij}^{\mathbf{\Theta}} = \{\rho_{ij}^{\mathbf{W}}, \rho_{ij}^{\boldsymbol{\mu}}, \rho_{ij}^{a}\}$ is the set of auxiliary variables for the parameters.

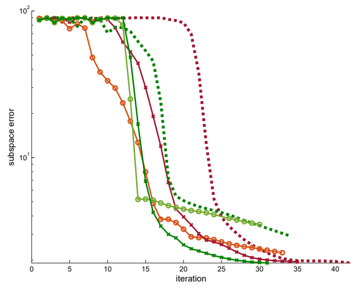
The augmented Lagrangian applying the proposed ADMM-NAP is similar to D-PPCA except that η becomes η_{ij} with λ_i, γ_i, β_i being Lagrange multipliers for the PPCA parameters for node i. The adaptive penalty constraint η_{ij}^t controls the speed of parameter propagation dynamically so that the overall optimization empirically converges more quickly than the standard ADMM-based D-PPCA. One can solve this optimization using the distributed EM approach [36]. The E-step of the D-PPCA is the same as the centralized counterpart [11]. The M-step is similar to D-PPCA except we use separate η_{ij} for each edge. The update formulas for the three parameters are similar and provide an example update for $\boldsymbol{\mu}_i$. Once all the parameters and the Lagrange multipliers are updated, we update η_{ij} and $\mathcal{T}_{ij}$ using (4.39) and (4.40), respectively.

4.2.4 Application: Distributed Computer Vision

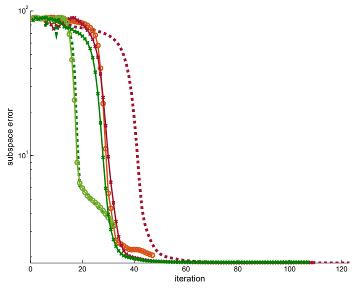
For evaluation, we use the similar set of experiments we used so far in the earlier chapters. We generate a noisy Gaussian dataset for the synthetic data experiments, and use the Caltech Turntable and the Hopkins155 on the distributed affine structure from motion problem for real data experiments.

Results on Synthetic Data : We first analyze and compare the proposed methods (ADMM-VP, ADMM-AP, ADMM-NAP, ADMM-VP + AP, ADMM-VP + NAP) with the baseline method, using synthetic data. Next, we apply our method to a distributed structure from motion problem using two benchmark real-world datasets. For the baseline, we compare with the ADMM-based D-PPCA denoted as ADMM with fixed penalty $\eta^t = \eta^0$. Unless noted otherwise, we used $\eta^0 = 10$. To assess convergence, we compare the relative change of (4.44) to a fixed threshold (10^{-3} in this case) for the D-PPCA experiments, as in Chapter 3.

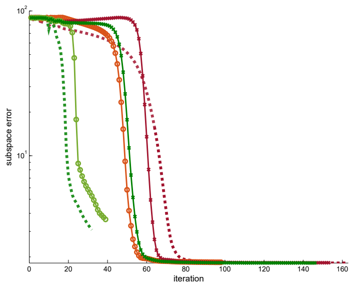
We generated 500 samples of 20-dimensional observations from a 5-dim subspace following $\mathcal{N}(\mathbf{0}, \mathbf{I})$, with the Gaussian measurement of noise following $\mathcal{N}(\mathbf{0}, 0.2 \cdot \mathbf{I})$. For the distributed settings, the samples are assigned to each node evenly. All experiments



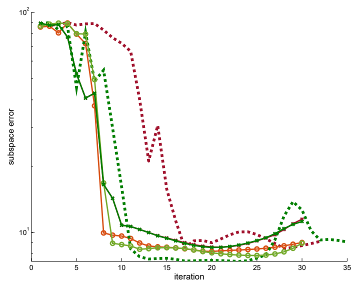
(a) 12 nodes (complete)



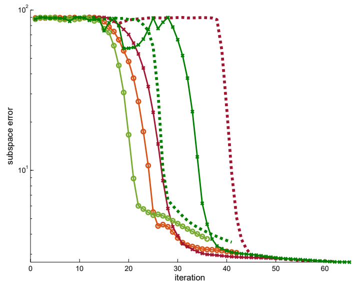
(b) 16 nodes (complete)



(c) 20 nodes (complete)



(d) 20 nodes (complete)



(e) 20 nodes (cluster)

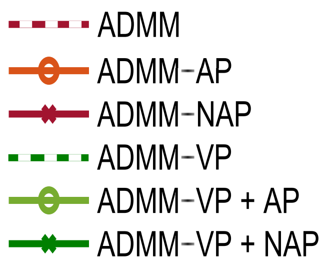


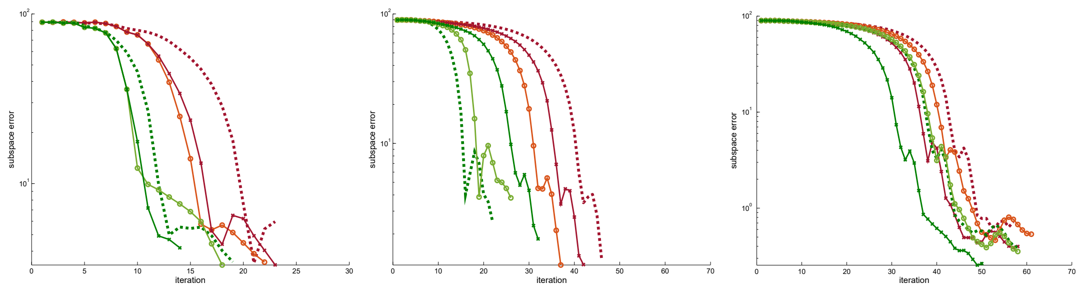
Figure 4.7: The comparison of proposed methods and the baseline ADMM using the subspace angle error of the projection matrix with (a-c) different graph size and (c-e) different network topology. Best viewed in color.

are run with 20 independent random initializations. We measured the number of iterations to convergence and the maximum subspace angle error versus the ground truth defined as the maximum of subspace angles between each node's projection matrix, and the ground truth projection matrix. We examined the impact of different graph topologies and different graph sizes. We tested three network topologies: complete, ring and cluster (a connected graph consists of two complete graphs linked with an edge). For the graph size, we tested on settings using 12, 16, and 20 nodes.

The top three plots in Figure 4.7 depict results over varying number of nodes while fixing the graph topology as the complete graph. We plot the median result out of the 20 independent initializations. We observed that the speed up with the proposed method, particularly for ADMM-VP and its variants, becomes more significant as the number of nodes increases. This suggests that the proposed method can be of particular use as the size of an application problem increases. Figure 4.7c to 4.7e show the performance in the context of different network topologies. Our proposed methods converge more quickly or at the same rate as the standard ADMM. In some cases, either the standard ADMM or our methods could converge to a local optima, e.g., some of our methods in Figure 4.7c prematurely converged, however, they still have very good performance that is less than 2 degrees of subspace angle. The proposed method works most robustly in the complete graph setting. In other words as the graph connectivity increases, the convergence property of the proposed method improves. Note also that ADMM-VP works best in the complete graph while ADMM-AP / NAP are better than the ADMM-VP in weakly connected networks (e.g., a ring which exhibits the sparsest connectivity resulting in long (error) propagation effects and, subsequently, a great deal of variable behavior). This makes sense as ADMM-VP depends on residual computation, and the proposed local residual computation becomes less accurate compared to the complete graph when the global residual can be computed.

Results on Real Data : We tested the performance of our method on five objects of Caltech Turntable [64] and Hopkins 155 [66] datasets, as in [1]. The goal here is to jointly estimate the 3D structure of the objects as well as the camera motion, however in a distributed camera network setting. The input measurement matrix is defined as $2 \times F$ by N, where F denotes the number of frames and N denotes the number of points. By applying PCA, we can decompose the input into the camera pose $\mathbf{W}_i$ and the 3D structure $\mathbb{E}[\mathbf{z}_{in}], n = 1..N_i$. For the detailed experimental setting, refer to [60, 1]. As a performance measure, we used the maximum subspace angle error versus the centralized SVD-reconstructed structure. The network setting assumes five cameras on a complete graph.

Figure 4.8 shows the result on the Caltech Turntable dataset. First, we compare Figure 4.8a and 4.8b. One can see that when the graph is less connected (Figure 4.8a), the proposed adaptive penalty method can boost ADMM-VP which cannot utilize the full residual information of a fully connected case (Figure 4.8b), as explained in synthetic data experiments. Next, we compare Figure 4.8b and 4.8c. The network topologies are the same (complete) but the $t^{\max}$ value required for ADMM-VP, ADMM-AP, and ADMM-VP + AP is different in these two groups of experiments. When $t^{max} = 50$ (Figure 4.8b), all methods can accelerate throughout the iterations. However, when



(a) $t^{max} = 50$ (ring) (b) $t^{max} = 50$ (complete) (c) $t^{max} = 5$ (complete)

Figure 4.8: The comparison of proposed methods and the baseline ADMM using the subspace angle error of the reconstructed 3D structure with one object in the Caltech dataset (Standing). See Figure 4.7 for plot labels. Best viewed in color.

$t^{max} = 5$ (Figure 4.8c), the methods that depend on t^{max} cannot accelerate after 5 iterations, thus showing behavior similar to the baseline ADMM. On the other hand, ADMM-NAP-based methods can accelerate by adaptively modifying the maximum number of penalty updates. Note that one can choose any small value of $\mathcal{T}$, and $\mathcal{T}_{ij}$ is increased automatically using (4.40). More results on the other four objects in the Caltech dataset can be found in supplementary materials of [4].

For the Hopkins 155 dataset, we compared methods on 135 objects using the same approach as [1]. For each method considered, we computed the mean number of iterations until convergence. Since some objects in the dataset are point trajectories of non-rigid structure, it is inevitable for simple linear models to fail for those objects. Thus, we omitted objects which yielded more than 15 degrees when calculating the mean. For each object, we tested 5 independent random initializations. For ADMM-AP, ADMM-NAP and ADMM-VP + NAP, we found no significant speed up over the baseline ADMM. For ADMM-VP and ADMM-VP + AP, we could obtain 40.2% and 37.3% speed up, respectively, if we use the complete network. In the ring network, the amount of improvement becomes smaller. This small or no improvement of speed is mainly due to the fact that the baseline ADMM converges rapidly enough (typically < 100 iterations), thus there is little room for the proposed methods to speed up the optimization. As observed from the synthetic experiments and the Caltech dataset, the acceleration of the proposed methods occurs at the earlier iterations of the optimization.

Thus, if one can develop a better convergence checking criterion depending on the application, the proposed methods can be a very viable choice due to their parameter-free nature.

4.3 Summary

In this chapter, we introduced two extensions of our distributed probabilistic learning framework. In the first part, we introduced a general approximate inference approach using MFVI for learning parameters of traditional centralized probabilistic models in a distributed setting. The main idea is to split the data into different nodes, impose consensus constraints on the posterior parameters of each node, and solve the constrained variational inference using Bregman ADMM. We illustrated this approach with D-BPCA for distributed SfM applications. Experimental results showed that Bayesian approaches show substantial and consistent improvements over the traditional maximum likelihood-based approach, with an additional benefit of it being a natural way to devise an online learning framework.

In the second part, we introduced novel adaptive penalty update methods for ADMM that can be applied to consensus distributed learning frameworks. Contrary to previous approaches, our adaptive penalty update methods, ADMM-AP and ADMM-NAP do not depend on parameters that require manual tuning. Using both synthetic and real data experiments, we showed the empirical effectiveness of the proposed methods over the baseline. In addition, we found that the performance of ADMM-VP decreases with weakly connected graphs, and in those cases, ADMM-AP and ADMM-NAP can be useful. These methods do leave some room for improvements. For the problems when the standard ADMM can converge quickly enough, the proposed methods may show less than significant gains. A better convergence criterion may help stop the proposed algorithms at earlier numbers of iterations (e.g., a criterion that can stop algorithms to remove long tails in Figure 4.7b or 4.7c would be useful).

Chapter 5

Applications in Multi-Agent Trajectory Estimation

In this chapter, we consider an application of the consensus-based distributed learning framework. So far, we only considered the problem in terms of visual sensor networks, i.e., with cameras being the nodes of the network, then imposed the consensus constraints on *model parameters estimated* by the cameras. Now, we will consider another type of consensus imposed on a different aspect of each camera's local information, i.e., a consensus-based *state estimation* problem. To see this, we consider the problem of crowd trajectory estimation as a global optimization of multiple agents.

5.1 Crowd Trajectory Estimation

The modelling, simulation, and analysis of crowds [117] has received widespread interests from many disciplines, including computer vision and graphics. Tracking of the movement of individuals in crowds is an indispensable component to crowd modelling and analysis, with applications in surveillance, crowd management, security and disaster prevention, crowd evacuation studies, as well as data-driven animation for visual effects and games. Crowd tracking is uniquely challenging since tracking must often be performed over large spatial areas, where multiple sensors need to be used to obtain sufficient coverage. However, there are several challenging issues to robustly track trajectories or even reconstructing human motion from tracked tracklets (partial trajectories) in multiple stages of the tracking process. We start our discussion by identifying what the crowd tracking problem is, and potential challenges.

The crowd tracking problem, in terms of scale, can be divided into three groups: macroscopic, mesoscopic, and microscopic approaches [118, 119]. The macroscopic approaches [120] track very dense crowds when each individual is not easy to identify.

These approaches typically consider the *flow* of a crowd and are usually applied on highly dense crowd movement simulations. The mesoscopic approaches are used for tracking groups of people. These approaches consider groups as crowd *blobs*. The microscopic approaches [121] consider the case when individuals can be identified and tracked separately. In this case, the crowd tracking problem becomes a multi-target object, or specifically a pedestrian tracking problem. An extensive amount of research on this problem has been carried out within the computer vision community. Among them, methods based on human behavior modeling, starting from a variant of Discrete Choice Model [122] or Social Force (SF) [123] models, have been proposed with applications in robotics, e.g., [124]. Our work is complementary in nature and seeks to address the problem of reconstructing crowd movement, tracked using multiple, noisy sensors.

The microscopic crowd tracking problem, using multiple visual sensors, is an open problem and a very active area of research. It starts with object detection and tracking at the local sensor level, then tracklets obtained from sensors are collected into a central processing node (or server). At this level, tracklets of the same subject are linked to build the global trajectory of the target. Typical crowd (or any object) trackers takes an iterative refinement approach, i.e., first estimate the global trajectory based on available tracklets, then refine the detection results of individual points within tracklets based on the estimated global trajectory, and reiterate the global estimation process.

However, current iterative refinement-based crowd tracking methods encounter challenges in multiple, large camera network settings. We identify three major issues here. First, the information provided by these visual sensors (or the results of the local tracking algorithms) may be noisy, and the combination of the sensors may not have complete coverage, resulting in missing or incomplete traces. Second, as the motivating example mentioned in Chapter 1, real-time processing is crucial in crowd tracking, and in large camera networks, running time-consuming centralized iterative processing may not result in a timely output. Third, the tracklet matching process described above is a well known, challenging problem, and doing this in a camera network setting is another important problem that is not trivial to solve, although some correspondence matching methods for distributed cameras have been proposed [125].

To this end, in this chapter, we focus on the first two issues, i.e., we propose a consensus-based, distributed, global optimization framework that can reconstruct the holistic crowd motion from multiple, noisy sensors, with incomplete coverage. We formulate the problem as follows: Consider tracked information from multiple visual sensors, with incomplete coverage, where sensor observations may be arbitrarily noisy. We assume that we get tracklets from each sensor, using existing pedestrian tracking solutions. We assume that the traces of individuals in the crowd across sensors have correspondence which can be uniquely identified, using existing methods (e.g., using the Hungarian algorithm). The outcome trajectories often have missing or noisy information due to sensor inaccuracies, algorithmic (tracking or matching) failure, or insufficient coverage due to the limited field of view (FOV) of the sensors. Our goal is to reconstruct the holistic view of microscopic crowd motion, and we want the reconstruction to be robust to noise and missing tracklets, while satisfying the properties desired for most people.

To address the challenges described above, we propose a global optimization-based trajectory refinement framework using ADMM. In particular, our framework leverages a recently introduced message passing optimization method for multiple multi-agent trajectory planning [12]. We formulate our global optimization problem as two input positions (initial and goal) per person (agent), with modular energy functions to minimize. These energy functions encode various desirable properties of individuals in a crowd, including physical constraints (e.g., maximum possible speed for a person), social constraints (e.g., efficient travel path, avoid colliding with others) or environmental constraints (e.g., avoid colliding with static obstacles such as walls). The overview of the entire process is depicted in Figure 5.1.

The efficacy of our method relies on two central contributions. First, we propose a global trajectory refinement method, based on ADMM [15], for crowd trajectory estimation, that is inherently suitable to handle noisy and missing information. To the best of our knowledge, we are the first to extend and apply the ADMM-based method for the crowd trajectory estimation and refinement problem in this manner. Second, we

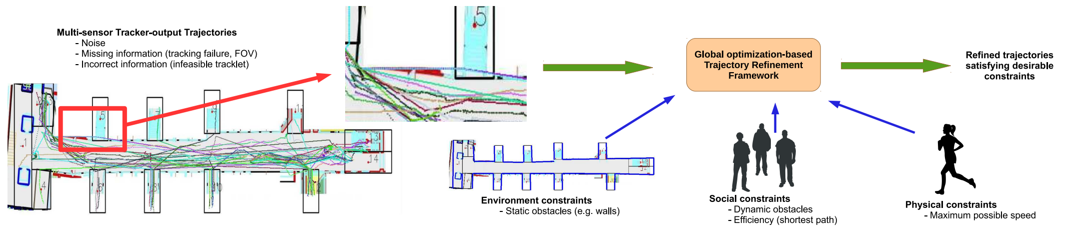

Multi-sensor Tracker-output Trajectories
- Noise
- Missing information (tracking failure, FOV)
- Incorrect information (infeasible tracklet)
Global optimization-based Trajectory Refinement Framework
Refined trajectories satisfying desirable constraints
Environment constraints
- Static obstacles (e.g. walls)
Social constraints
- Dynamic obstacles
- Efficiency (shortest path)
Physical constraints
- Maximum possible speed

Figure 5.1: Overview of proposed global optimization-based trajectory refinement framework

formulate the crowd trajectory estimation problem as a modular objective global optimization problem using a set of modular energy terms that encode physical, social, and environmental constraints. We validate our framework on synthetic data with increasing amounts of noisy and missing information, showcasing significant reconstruction improvements over baseline methods.

In the following section, we first briefly review related prior work on the crowd trajectory estimation problem. Next, we formally define the problem and show how it can be understood as a consensus-based distributed optimization problem. Then we explain how this formulation can be converted into the proposed global optimization-based trajectory estimation framework. Then, we show our experimental results to demonstrate that our framework can smooth out trajectories with noise and missing tracklets. Finally, we discuss limitations of our approach and potential future extensions.

5.2 Related Work

There is an extensive amount of research in the modelling, simulation, and analysis of crowds [126, 117, 127] at varying scales that capture both the macroscopic and microscopic aspects of crowd flow. We refer readers to comprehensive surveys in crowd tracking [118, 119], and focus on trajectory estimation for individuals in crowds.

As a precursor to pedestrian tracking, it is common to provide (or learn) a motion prior, which can be used to guide and improve tracking accuracy, which has been addressed from different perspectives [128, 120, 129]. A notable contribution is the work in [121, 130], which introduces the popular Social Forces model [131] to model the dynamic social behavior of individuals in a crowd, to serve as motion prior in order to improve tracking accuracy. This work is extended in [132], by proposing a multi-target tracking model that succeeds in uniformly including different sources of information. They include appearance, physical constraints, and the social behavior of walking people, and are amenable to further extensions. The model is built using a Conditional Random Field framework. As the model cannot be globally optimized, it adopts an approximate inference strategy.

Bera and Manocha [133, 134] introduced a real-time algorithm for trajectory estimation in medium-density crowds using adaptive particle filtering, while relying on a multi-agent motion model based on velocity obstacles [135]. The central idea behind their approach is to separate tracking from waypoint estimation, and utilize a motion model for estimation. This work is extended in the AdaPT [136] framework to provide real-time adaptive tracking for crowded scenes. In comparison to model-based approaches [121, 134] which rely on a synthetic simulator to provide motion priors for crowd tracking, our approach estimates crowd movement directly from the data by enforcing constraints on the estimated trajectories.

Alahi, et al. [137] proposed a global optimization approach with origin-destination prior, along with a novel social affinity feature map (SAM) that encodes behavioral patterns of a group of pedestrians learnt from large pedestrian data, and hence helps in the association of broken tracklets. We are interested in different types of behavioral patterns, which are characterized by the overall environment and a much wider range of groups of people, which may not necessarily be within the vicinity of one another.

Due to the inherent limitations of single camera trackers when applied to dense crowds, new approaches attempt to fuse the data from multiple cameras. The work in [138] alleviates the single camera tracking problem by using multiple overlapping camera sensors, and are able to handle severe and persistent occlusions in dense, crowded scenes. Our motivations are different, and seek to address the problem of multi-sensor crowd tracking with missing and incomplete information. The work in [119] uses a modified SF model to estimate movement in unobserved areas for target re-identification. For tracking and activity recognition works for camera networks in computer vision including tracking by consensus-based methods [63], we refer readers to a survey [60] and references therein.

Complementary to the crowd trajectory estimation problem, which is the focus of this chapter, is the simulation of crowd movement, with many proposed solutions having been published by the graphics community [127]. As described previously, crowd tracking, trajectory estimation and simulation are tightly coupled, where crowd simulators [131, 135] may be used as motion priors to improve tracking accuracy, and the output of crowd trackers may be used to train data-driven models for crowd simulation [139].

5.3 Global Optimization-based Trajectory Estimation

We now define the trajectory estimation problem and show how it can be considered as a consensus-based optimization problem, then as a global optimization problem.

5.3.1 Notaion and Problem Definition

Consider sets of 2D trajectories $\mathcal{D}$. As a general notation for such sets, we use $\mathcal{D} = \{(\mathbf{O}_i, l_i), \mathbf{Z}\}$, $\forall i \in (1, M)$, where M denotes total number of trajectories in the set. The trajectories $\mathbf{O}_i$ are derived from multiple sensors using pedestrian trackers. Since we have multiple views of the trajectories, essentially these are tracklets without identity $l_i \in \{1, ..., P\}$ of the agent whose trajectory is obtained from, where P is the maximum number of unique agents in $\mathcal{D}$. Therefore, we need to solve the reidentification (or mapping) problem over multiple cameras by using, e.g., [119]. Still, our trajectory may include noise and potentially missing information. Each

$$\mathbf{O}_i = \{(x_i^t, y_i^t, u_i^t)\}, \quad \forall t \in (1, N_i), \quad x_i^t, y_i^t \in \mathbb{R}$$

denotes a set of track points consisting of observed horizontal and vertical positions of the person of i-th trajectory at time t. N_i denotes the number of track points available for the trajectory i. As a notational shortcut, we denote the track point at time t for i-th trajectory as $\mathbf{o}_i^t = (x_i^t, y_i^t)$. To represent the potential uncertainty of information accompanied with each $\mathbf{t}_i$, we use $u_i^t \in [0, 1]$, where 0 means uncertain or missing and 1 means absolutely certain. This uncertainty measure can be a real value in $[0,1]$ estimated by the tracker, or simply a binary variable indicating the tracker has observed the point or it is missing.

$$\mathbf{Z} = \{(z^t_{k,x_1}, z^t_{k,x_2}, z^t_{k,y_1}, z^t_{k,y_2})\}, \quad \forall k = (1, N_k)$$

denotes the environmental configuration that $\mathcal{D}$ was observed where z^t_{k,x_1}, z^t_{k,x_2} denote horizontal minimum and maximum bound of k-th linear obstacle at time t, and z^t_{k,y_1}, z^t_{k,y_2} are defined similarly for the vertical dimension. All obstacles in the environment are considered as a combination of these linear obstacles. For example, a rectangular wall is a set of four linear obstacles. While it is possible to consider dynamic obstacles with this definiton, we consider static obstacles only. Similary to $\mathbf{o}^t_i$, we define $\mathbf{z}^t_k = (z^t_{k,x_1}, z^t_{k,x_2}, z^t_{k,y_1}, z^t_{k,y_2})$.

As mentioned in the earlier section, we assume that the tracklet matching (reidentification) can be done fairly well with available methods, with some potential room for refinement, which is the focal point of the proposed method. Note that since we assume that the tracklet matching is known, it is possible to omit l_i from $\mathcal{D}$ so that there is only one trajectory for each individual, by linking all tracklets in $\mathcal{D}$ belonging to the same individual. In this case, the index i denotes the index of an individual, i.e., $i \in \{1, ..., P\}$.

The goal here is to reconstruct and refine $\mathbf{T}_i$, i.e., the true *state* of agent i, which is essentially the set of 2D coordinates $\mathbf{t}^t_i = (\hat{x}^t_i, \hat{y}^t_i)$ on a calibrated plane common to all sensors. In other words, we consider the camera observed, and tracker output $\mathbf{o}^t_i$ as a *noise-corrupted representation* of $\mathbf{t}^t_i$. One can consider an analogy of $\mathbf{t}^t_i$ as the latent *state* variable $\mathbf{z}_{in}$ and $\mathbf{o}^t_i$ as the observation $\mathbf{x}_{in}$ we discussed in previous chapters, although we do not explicitly model parameters (θ in previous chapters), and the samples are temporally linked. We make the following assumptions: (1) The environmental configuration (obstacles) is known and static, (2) The true initial $(\hat{x}^1_i, \hat{y}^1_i)$ and estimated goal positions $(\hat{x}^{N_i}_i, \hat{y}^{N_i}_i)$ for each trajectory are known and reliable, (3) The identities of trajectories tracked by trackers are known, although it is possible that multiple parts of a trajectory can be missing. The identities across sensors are known.

5.3.2 Consensus-based Problem Formulation

Suppose we have J sensors in the surveillance environment $\mathbf{Z}$ we are monitoring. Let $\mathcal{D}^m$ be the set of trajectories observed by m-th sensor, where $m = 1..J$. For all $\mathcal{D}^m$, there are P pedestrian trajectories, although not all sensors have all trajectory information, obviously. For those missing parts, corresponding $u_i^t = 0$. Now, the goal of the global crowd *state* estimation problem we described in the previous section can be considered as the following consensus-based optimization problem

$$\begin{aligned} \arg\min_{\mathcal{T}} \quad & f(\mathcal{T}|\mathcal{D}) = \sum_{m=1}^{J} f(\mathcal{T}^m|\mathcal{D}^m) \\ s.t. \quad & \mathbf{T}_i^m = \boldsymbol{\tau}_{ij}^{mn}, \quad \boldsymbol{\tau}_{ij}^{mn} = \mathbf{T}_j^n, \quad \forall m, n = 1..J, m \neq n \end{aligned} \tag{5.1}$$

where $\mathbf{T}_i^m$ and $\mathbf{T}_j^n$ are state trajectories of observed trajectories $\mathbf{O}_i^m$ and $\mathbf{O}_j^n$ such that $\mathbf{O}_i^m \in \mathcal{D}^m$, $\mathbf{O}_j^n \in \mathcal{D}^n$, $l_i = l_j$. We use $\mathcal{T}^m = \{\mathbf{T}_i^m, i = 1..P\}$ to denote local copy of global state trajectory set $\mathcal{T}$ for the m-th node. Restating in words, this means that the estimated state trajectories should be the same across different sensors if they are from the same identity (i.e., agent or pedestrian). Note that the state trajectories are *not* necessarily in the measurement space (i.e., image frame coordinates). These trajectory points reside in the calibrated world coordinate that is shared (and known) across the sensors. This is not too strong an assumption if the sensors are static and pre-installed, which is typical in surveillance scenarios. The cost function $f(\cdot)$ can be a collection of different prior constraints that are desirable for optimal state trajectories as

$$f(\mathcal{T}^m|\mathcal{D}^m) = \sum_{i=1}^{P} f_p(\mathbf{T}_i^m|\mathcal{D}^m) + \sum_{i=1}^{P} f_e(\mathbf{T}_i^m|\mathbf{Z}) + \sum_{\forall(i,j), i\neq j} f_s(\mathbf{T}_i^m, \mathbf{T}_j^m) \tag{5.2}$$

where f_p denotes a unary constraint that only depends on individual trajectory, f_e denotes the binary constraint that depends on each trajectory against the environmental obstacles, and f_s denotes the social constraints between different agents. We will describe these constraints in detail below.

5.3.3 Constraints for Optimal Multi-Agent Trajectory

Given the trajectory data extracted from multiple sensors above, the goal is to transform $\mathbf{O}_i^m \to \mathbf{T}_i^m, \forall i$ which minimizes noise and uncertainity. We consider a model that each track point of each agent (pedestrian) is determined by minimizing the summation of energy functions that defines constraints on prior knowledge of the human trajectories. For the prior terms, we drop superscript m from $\mathbf{t}_i^t$ for notational brevity. This should not incur confusion as the prior terms are defined within same node.

Tracker output : Since our goal is to refine the given tracker output, it is reasonable to take current trajectory estimates with high confidence into account. To do this, we consider Gaussian prior over the initial tracker output on estimated trajectories as

$$E_{gt}(\mathbf{t}_i^t|\mathcal{D}^m) = u_i^t\|\mathbf{t}_i^t - \mathbf{o}_i^t\|^2 \tag{5.3}$$

where $\mathbf{o}_i^t$ is the tracker output and u_i^t acts as a precision parameter. Note that if the point is missing, i.e., $u_i^t = 0$, this prior is disabled.

Kinetic energy : As a reasonable assumption, we consider agents are trying to reach the goal position as soon as possible following the minimum travel distance. This constraint can be encoded as kinetic energy, i.e.,

$$E_{kn}(\mathbf{t}_i^t, \mathbf{t}_i^{t+1}) = c_{kn}\|\mathbf{t}_i^{t+1} - \mathbf{t}_i^t\|^2 \tag{5.4}$$

where c_{kn} is the expected mass of one agent. We typically set $c_{kn} = 1$. If we sum up all E_{kn} for an agent, that is the total distance agent i travelled along the trajectory.

Physical constraint - Maximum velocity : Physical constraints of the human body put limits on the maximum walking speed of an agent. Thus, we can consider a constraint-only function, that encodes this maximum possible distance an agent can travel within unit time frame as

$$E_{mv}(\mathbf{t}_i^t, \mathbf{t}_i^{t+1}) = \begin{cases} 0 & \text{if } \|\mathbf{t}_i^{t+1} - \mathbf{t}_i^t\| \leq c_{mv}, \\ \infty & \text{otherwise.} \end{cases} \tag{5.5}$$

This sets an upper bound on the velocity of agent movement with c_{mv} of which value is decided by the frame rate of the trajectory and physical constraint of an average human. Note that unlike [12], we do not consider minimum velocity bound, since humans can stop time to time during their travel.

Social constraint - Collisions with other agents : Collisions are typically what we want to avoid. This is particularly important as initial tracker output can easily contain collisions between trajectories, even though there was no actual collision due to occlusions.

As in the maximum velocity case, we consider a constraint function as

$$C_s(\mathbf{t}_i^t, \mathbf{t}_j^t, \mathbf{t}_i^{t+1}, \mathbf{t}_j^{t+1}, r_i, r_j) = \begin{cases} 0 & \text{if } \|\alpha(\mathbf{t}_i^{t+1} - \mathbf{t}_j^{t+1}) + (1-\alpha)(\mathbf{t}_i^t - \mathbf{t}_j^t)\| \\ & \quad \geq (r_i + r_j), \quad \forall \alpha \in [0,1] \\ \infty & \text{otherwise} \end{cases}$$

where r_i, r_j are radius of agents i and j. This function is minimized if the two agents i and j are at least $r_i + r_j$ apart at all times between t and $t+1$.

Environmental constraint - Collisions with walls : Another type of collisions is that between agents and the environment, i.e., obstacles. As mentioned, we consider static obstacles only. Let the two coordinates of linear obstacle k at time t as $\mathbf{z}_{k_1}^t = (z_{k,x_1}^t, z_{k,y_1}^t)$ and $\mathbf{z}_{k_2}^t = (z_{k,x_2}^t, z_{k,y_2}^t)$. Then, we can consider a constraint function for the wall collision as

$$C_e(\mathbf{t}_i^t, \mathbf{t}_i^{t+1}, r_i, \mathbf{z}_{k_1}^t, \mathbf{z}_{k_2}^t) = \begin{cases} 0 & \text{if } \|(\alpha\mathbf{t}_i^{t+1} + (1-\alpha)\mathbf{t}_i^t) - (\beta\mathbf{z}_{k_1}^t + (1-\beta)\mathbf{z}_{k_2}^t)\| \\ & \quad \geq r_i, \quad \forall \alpha, \beta \in [0,1] \\ \infty & \text{otherwise} \end{cases}$$

thus the agent i will not cross the linear obstacle at all times between t and $t+1$. As one can see, C_s and C_e has the useful relationship that

$$C_s(\mathbf{t}_i^t, \mathbf{t}_j^t, \mathbf{t}_i^{t+1}, \mathbf{t}_j^{t+1}, r_i, r_j) = C_e(\mathbf{t}_j^t - \mathbf{t}_i^t, \mathbf{t}_j^{t+1} - \mathbf{t}_i^{t+1}, r_i + r_j, 0, 0) \tag{5.6}$$

so if we know how to solve C_e, C_s can be solved by converting it into C_e. For details on this proximal operator, refer to [12].

5.3.4 Combined Global Objective Formulation and Optimization

For each i-th agent, the three partial objective terms in (5.2) can be defined as

$$f_p(\mathbf{T}_i^m|\mathcal{D}^m) = \sum_{t=1}^{N_i} E_{gt}(\mathbf{t}_i^t|\mathcal{D}^m) + \sum_{t=1}^{N_i-1} E_{kn}(\mathbf{t}_i^t, \mathbf{t}_i^{t+1}) + \sum_{t=1}^{N_i-1} E_{mv}(\mathbf{t}_i^t, \mathbf{t}_i^{t+1}) \tag{5.7}$$

$$f_e(\mathbf{T}_i^m|\mathbf{Z}) = \sum_{t=1}^{N_i} \sum_{\forall(i,k)} C_e(\mathbf{t}_i^t, \mathbf{t}_i^{t+1}, r_i, \mathbf{z}_{k_1}^t, \mathbf{z}_{k_2}^t) \tag{5.8}$$

$$f_s(\mathbf{T}_i^m, \mathbf{T}_j^m) = \sum_{t=1}^{N_i} \sum_{\forall(i,j),i\neq j} C_s(\mathbf{t}_i^t, \mathbf{t}_j^t, \mathbf{t}_i^{t+1}, \mathbf{t}_j^{t+1}, r_i, r_j) \tag{5.9}$$

where $\mathbf{t}_i^1, \mathbf{t}_i^{N_i}$ are given and fixed. Thus, we only determine $\mathbf{t}_i^2, \cdots, \mathbf{t}_i^{(N_i-1)}$. Note that when optimizing $\mathbf{T}_i^m$, all $\mathbf{T}_j^m \in \mathcal{D}^m$ that may have the possibility of collision with $\mathbf{T}_i^m$ should be considered as well due to C_s. For each m-th local node, we minimize all trajectories available in the node, thus the local objective, by minimizing

$$\operatorname*{arg\,min}_{\mathcal{T}^m} f(\mathcal{T}^m|\mathcal{D}^m) = \sum_{i=1}^{P} f_p(\mathbf{T}_i^m|\mathcal{D}^m) + \sum_{i=1}^{P} f_e(\mathbf{T}_i^m|\mathbf{Z}) + \sum_{\forall(i,j),i\neq j} f_s(\mathbf{T}_i^m, \mathbf{T}_j^m). \tag{5.10}$$

Recall that our global objective (5.1) is

$$\begin{aligned} &\operatorname*{arg\,min}_{\mathcal{T}^m,\forall m\in\mathcal{V}} \quad \sum_{m=1}^{J} f(\mathcal{T}^m|\mathcal{D}^m) \\ &\qquad s.t. \quad \mathbf{T}_a^m = \boldsymbol{\tau}_{ab}^{mn}, \quad \boldsymbol{\tau}_{ab}^{mn} = \mathbf{T}_b^n, \quad \forall m,n = 1..J, m\neq n \end{aligned} \tag{5.11}$$

where $\mathbf{T}_a^m$ and $\mathbf{T}_b^n$, $l_a = l_b$. We can consider the augmented Lagrangian as

$$\begin{aligned} \operatorname*{arg\,min}_{\mathcal{T}^m,\forall m\in\mathcal{V}} \quad \sum_{m=1}^{J} \Bigg[& f(\mathcal{T}^m|\mathcal{D}^m) + \sum_{\forall(a,b),l_a=l_b} \left\{ \boldsymbol{\lambda}_{ab1}^\top (\mathbf{T}_a^m - \boldsymbol{\tau}_{ab}^{mn}) + \boldsymbol{\lambda}_{ab2}^\top (\boldsymbol{\tau}_{ab}^{mn} - \mathbf{T}_b^n) \right\} \\ & + \sum_{\forall(a,b),l_a=l_b} \left\{ \frac{\eta}{2} \|\mathbf{T}_a^m - \boldsymbol{\tau}_{ab}^{mn}\|^2 + \frac{\eta}{2} \|\boldsymbol{\tau}_{ab}^{mn} - \mathbf{T}_b^n\|^2 \right\} \Bigg]. \end{aligned} \tag{5.12}$$

It is important to note the notational difference between (i,j) and (a,b) that the index pair (i,j) is the pair in (5.10) between trajectories within a sensor data $\mathcal{T}^m$, while the index pair (a,b) in (5.11) denotes the pair between trajectories across two sensor data $\mathcal{T}^m$ and $\mathcal{T}^n$, $m \neq n$. However, we will see that these terms can be expressed using one index notation i. To see this, we expand the cost $f(\cdot)$ as

$$\begin{aligned} \operatorname*{arg\,min}_{\mathcal{T}^m,\forall m\in\mathcal{V}} \quad \sum_{m=1}^{J} \Bigg[& (\cdots) + \sum_{\forall(i,j),i\neq j} f_s(\mathbf{T}_i^m, \mathbf{T}_j^m) \\ & + \sum_{\forall(a,b),l_a=l_b} \left\{ \boldsymbol{\lambda}_{ab1}^\top (\mathbf{T}_a^m - \boldsymbol{\tau}_{ab}^{mn}) + \boldsymbol{\lambda}_{ab2}^\top (\boldsymbol{\tau}_{ab}^{mn} - \mathbf{T}_b^n) \right\} \\ & + \sum_{\forall(a,b),l_a=l_b} \left\{ \frac{\eta}{2} \|\mathbf{T}_a^m - \boldsymbol{\tau}_{ab}^{mn}\|^2 + \frac{\eta}{2} \|\boldsymbol{\tau}_{ab}^{mn} - \mathbf{T}_b^n\|^2 \right\} \Bigg] \end{aligned} \tag{5.13}$$

where we suppressed the unary and environmental constraints to focus on the social constraint term. There are two important observations here. One is that $\mathbf{T}_a^m \in \mathcal{T}^m$, $\mathbf{T}_i^m \in \mathcal{T}^m$ and $\mathbf{T}_j^m \in \mathcal{T}^m$. The other is that the trajectories $\mathbf{T}_a^m$ and $\mathbf{T}_b^n$ have the condition that $l_a = l_b$ while $l_i \neq l_j$. This means that every trajectory in $\mathcal{T}^m$ should have the penalty terms imposed, and essentially, the indexing of i and a are for independent summation terms, thus we can use the same index i for all summations. Therefore, we can rewrite (5.13) as

$$\begin{aligned}\underset{\mathcal{T}^m, \forall m \in \mathcal{V}}{\arg\min} \quad \sum_{m=1}^{J} \Bigg[(\cdots) &+ \sum_{\forall (i,j), i \neq j} f_s(\mathbf{T}_i^m, \mathbf{T}_j^m) \\ &+ \sum_{\forall (i,k), l_i = l_k} \left\{ \boldsymbol{\lambda}_{ik1}^\top (\mathbf{T}_i^m - \boldsymbol{\tau}_{ik}^{mn}) + \boldsymbol{\lambda}_{ik2}^\top (\boldsymbol{\tau}_{ik}^{mn} - \mathbf{T}_k^n) \right\} \\ &+ \sum_{\forall (i,k), l_i = l_k} \left\{ \frac{\eta}{2} \|\mathbf{T}_i^m - \boldsymbol{\tau}_{ik}^{mn}\|^2 + \frac{\eta}{2} \|\boldsymbol{\tau}_{ik}^{mn} - \mathbf{T}_k^n\|^2 \right\} \Bigg], \end{aligned} \tag{5.14}$$

where $\mathbf{T}_k^n \in \mathcal{T}^n$ and of course $m \neq n$. Since we assumed that (copies of) all state trajectory information is available in each sensor, all summations using index i are the summation of the terms with i in range of $1..P$. It can be seen clearly now that the global objective (5.14) can be computed in a distributed way, as we have done in previous chapters.

However, the direct minimization of the global objective (5.14) is not easy. Particularly, collision constraint terms f_e and f_s are nonconvex. To solve this, we utilize a message passing interpretation of alternating direction method of multiplier [12, 13]. We provide a brief review on the message passing ADMM in Appendix D. Using the message passing ADMM, the global objective (5.14) can be solved as a consensus optimization problem of many different energy terms with a common set of latent variables, and which is the state trajectories $\mathbf{T}_i^m$ of all agents. One can consider this as additional consensus imposed both within each node in addition to the consensus across the nodes.

Specifically, we can consider the state trajectory $\mathbf{t}_i^t$ used in each energy term as a local copy of a globally shared trajectory $\overline{\mathbf{t}}_i^t$ which we initialize with tracker output. Our goal is to minimize (5.14) while imposing consensus constraint $\mathbf{t}_i^t = \overline{\mathbf{t}}_i^t$ for all $\mathbf{T}_i^m \in \mathcal{D}^m$. Let S be the total number of energy terms we need to minimize and each term is indexed by $s = 1..S$. We denote a set of terms that shares the same $\mathbf{t}_i^t$ as Θ_i^t. Let $\rho_{s,i}^t$ be the learning rate ρ in ADMM update for the optimization of term s with respect to $\mathbf{t}_i^t$. Similarly, we denote $\boldsymbol{\mu}_{s,i}^t$ as Lagrange multiplier of term s with respect to $\mathbf{t}_i^t$. To avoid confusion, we use $(\cdot)_s^t$ to denote local variable for term s at iteration t in the ADMM update below. The message passing ADMM can optimize each term, e.g., E_{gt}, by following the sequence of updates

$$\mathbf{n}_{s,i}^t = (\bar{\mathbf{t}}_i^t)_s^t - \boldsymbol{\mu}_{s,i}^t \tag{5.15}$$

$$(\mathbf{t}_i^t)_s^{t+1} = \arg\min_{\mathbf{t}_i^t} \left(u_i^t \|\mathbf{t}_i^t - \mathbf{o}_i^t\| + \frac{\rho_{s,i}^t}{2} \|\mathbf{t}_i^t - \mathbf{n}_{s,i}^t\|^2 \right) \tag{5.16}$$

$$\mathbf{m}_{s,i}^t = (\mathbf{t}_i^t)_s^{t+1} + \boldsymbol{\mu}_{s,i}^t \tag{5.17}$$

$$(\bar{\mathbf{t}}_i^t)_s^{t+1} = \frac{\sum_{\Theta_i^t} \rho_{s,i}^t \mathbf{m}_{s,i}^t}{\sum_{\Theta_i^t} \rho_{s,i}^t} \tag{5.18}$$

$$(\boldsymbol{\mu}_{s,i}^t)^{t+1} = (\boldsymbol{\mu}_{s,i}^t)^t + \left((\mathbf{t}_i^t)_s^{t+1} - (\bar{\mathbf{t}}_i^t)_s^{t+1}\right). \tag{5.19}$$

It is important to note that the squared penalty in update (5.16) must be applied to each of $\mathbf{t}_i^t$ in each energy term. If the energy term requires two or more variables, the number of squared penalty terms also matches to the number of variables. It is easy to see that the globally shared trajectory points $\bar{\mathbf{t}}_i^t$ will essentially become a weighted average of estimates from all energy terms that involves $\mathbf{t}_i^t$ in its local minimization. The overall optimization is iterative updates as above on all energy terms we introduced in the previous section. For details on local minimization of (5.16) for different types of energy terms, please refer to [12].

5.3.5 Discussions on Message Passing ADMM

A major benefit of the framework proposed in this chapter is that it can consider uncertainty or missing information from the tracker output, as explained in Section 5.3.3. This can be intuitively interpreted as disconnecting edges between the pair wise distance energy function E_{gt} for the missing point, and the function that aggregates the information from energy functions and imposes consensus constraint from message passing structure in [12], although the authors did not consider handling the missing values of

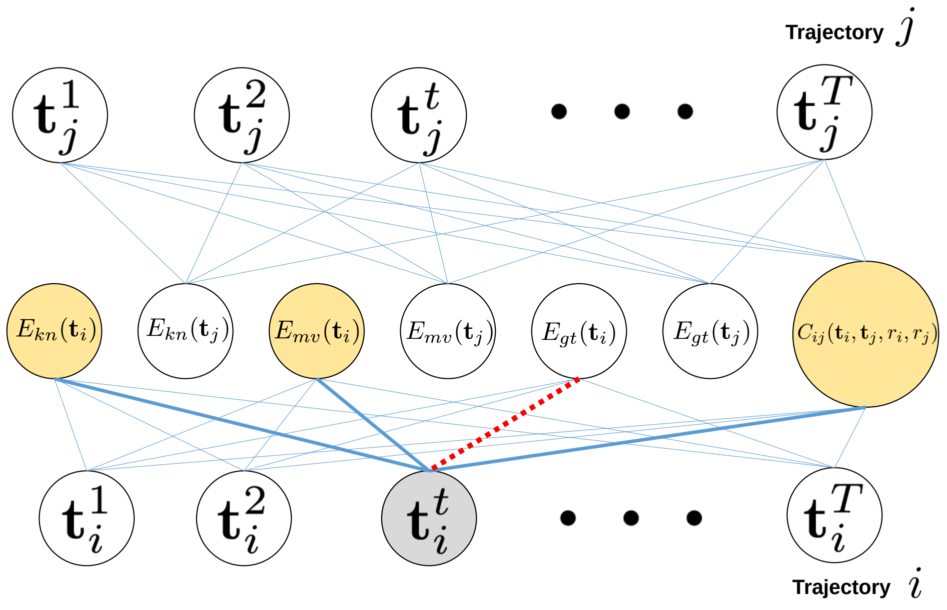
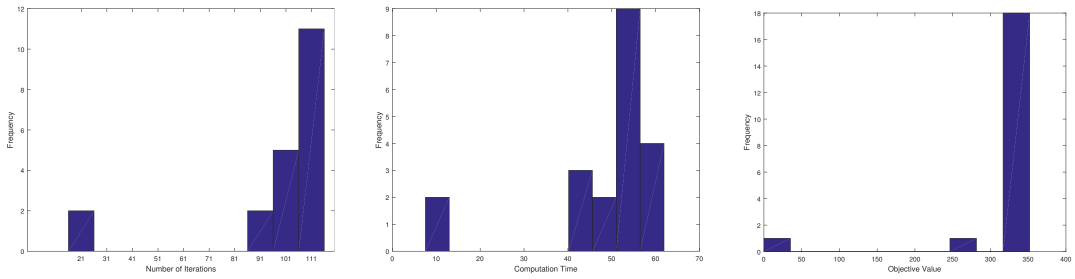


Figure 5.2: Graphical representation of our optimization framework given two trajectories i and j, assuming t-th point is missing in trajectory i. The connection of point $\mathbf{t}_i^t$ (shaded as gray) to energy term $E_{gt}(\mathbf{t}_i)$ is disconnected (red). Therefore, $\mathbf{t}_i^t$ will be estimated purely based on other terms (shaded as yellow). Note that τ does not appear in this figure since the prior terms are defined within a sensor, not across sensors.

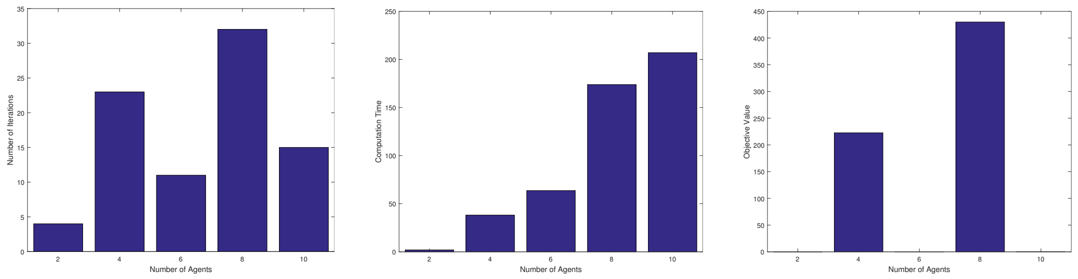
input data. Figure 5.2 shows an example graphical description of our missing value handling scheme.

One may wonder at this point about whether the distributed probabilistic models we introduced in previous chapters can utilize the message passing ADMM algorithm. While it is possible to apply the algorithm to the problems we discussed earlier, e.g., distributed probabilistic PCA, there is not much benefit over our proposed approach from previous chapters for two reasons. First, in this chapter, we employed the message passing ADMM because the derivation of closed form update for (5.14) was not easy, so we wanted to utilize the proximal operators provided by [12, 13]. For the problems in previous chapters, there is not much benefit of using off-the-shelf local optimizer, since it is possible to derive the closed form solution for the parameter updates. Second, the message passing ADMM is more like the consensus formulation of (2.20), i.e., parallel setting, unlike decentralized consensus formulation of (2.25), which we used throughout the thesis including the problem formulation part of this chapter until (5.14). It would be an interesting future direction to derive proximal operators using (2.25).



(a) # of iterations (b) Comput. Time (c) Obj. value

Figure 5.3: Stability experiments on ADMM algorithm with 2 agents, 5 break points (frames), and 20 random initializations. As can be seen, approximately 100 iterations and 50-60 seconds, the optimization is complete. Note that early stops yield premature results.



(a) # of iterations (b) Comput. Time (c) Obj. value

Figure 5.4: Scalability experiments on ADMM algorithms with 2, 4, 6, 8, 10 agents, 5 break points (frames), and 1 random initialization. As a typical trend, number of iterations and computation time increase almost linearly.

5.4 Evaluation

To demonstrate the utility of the proposed method, we conducted experiments on various standard synthetic crowd simulation scenario trajectories. We used the popular Social Forces model [131] to generate the simulation data.

5.4.1 Non-convex global optimization

ADMM, a convex optimization algorithm, is known to give reasonably good solution efficiently for many non-convex problems in practice. Here, we show the stability of the ADMM framework on a non-convex problem that is closely related to what we want to tackle. Following [12], we conducted experiments on the concentric circle (CONF1) scenario, in which agents will change their positions with agents at antipodal positions.

Figure 5.3 and 5.4 show CONF1 scenario experimental results with varying number of agents and initializations. We report number of iterations, computation time and objective values. As one can see, the proposed framework converges reasonably well in the majority of cases, although the problem is non-convex.

5.4.2 Robustness to missing tracklets and noise

To demonstrate the robustness of our framework to missing values and noise, we conducted experiments on synthetic crowd simulation scenarios. We considered 6 environment benchmarks, illustrated in Table 5.1. The trajectories of the crowd for each benchmark were generated synthetically using the Social Forces crowd simulation model [131]. We simulated 3000 frames per scenario using 30 - 40 agents, and then sub-sampled the trajectories so that the duration of the trajectory is approximately 100 frames.

Noisy Trajectories. Table 5.2 (left hand side) shows the reconstruction performance against the ground truth trajectory, with synthetically added noise. To demonstrate our reconstruction robustness to noise, we considered additive white Gaussian noise (AWGN) with different amounts of signal-to-noise ratio (SNR), ranging from 30, 50, and 70 dB. As a baseline comparison, we used a median filter. For noisy data experiments, we considered an additional baseline of *no-filter*, meaning that the original corrupted signal was without any noise removal process having been applied. We used root mean squared error for the measure. As shown in Table 5.2, the proposed method effectively improves the reconstruction performance over a naïve median filter by a significant margin. Table 5.3 and Table 5.4 visually compares the noisy trajectories (SNR 50 dB) and the corresponding reconstructed trajectories, using our method on the different environment benchmarks. Confirming our quantitative results in Table 5.2, the reconstructed trajectories are noticeably smoother, lack discontinuities, and are collision-free. Our method is thus able to effectively reconstruct trajectories while removing artifacts introduced as a result of noise.

Trajectories with Missing Information. Table 5.2 (right hand side) shows the reconstruction performance against the ground truth trajectory, where portions of the trajectory were artificially removed. We removed between 10 - 30 % of the trajectories of each agent to evaluate the robustness of our method. The qualitative results shown in Table 5.5 and Table 5.6, confirm our quantitative results. Our method is able to effectively reconstruct large portions of the missing information in the trajectories, while ensuring trajectories are artifact-free (without collisions and discontinuities), and still preserving the original essence of the crowd dataset.

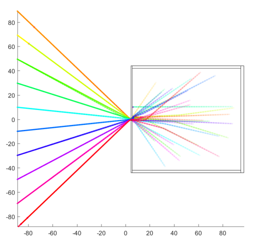
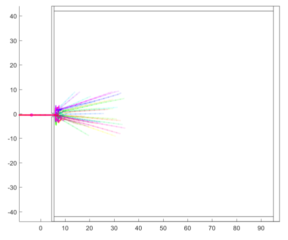
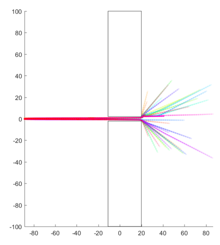
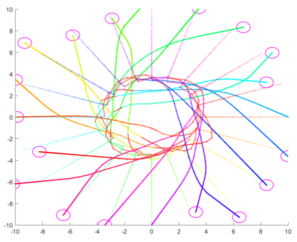
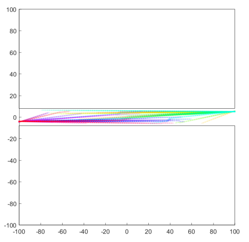
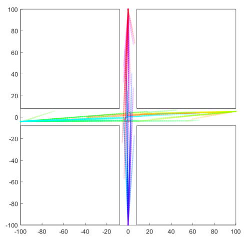
An interesting observation to note is that the proposed method shows strong reconstruction performance over the baseline method when congestion exists (e.g., bottleneck-evacuation-2 and bottleneck-squeeze, hallway-two-way). In case of bottleneck-evacuation, since the goal is far away from the exit, congestion is not heavy for the agents. For the hallway case, two-way has more chance to collide than four-way since we have the somewhat small number of agents of 32 (8 per way). This phenomenon naturally makes sense as the simulated trajectory will show complex turns when there is congestion, thus linear interpolation or simple median filtering is prone to failure.

Noise and Missing Information. As a more realistic case, we considered the situation when both the noise and missing information exist in the trajectories. Table 5.7 shows the result on some exemplary combinations of the cases. We picked the *bottleneck-evacuation* scenario, since our method found it most challenging. As shown in the table, the general trend is similar to Table 5.2, providing additional support for the robustness of the proposed method. Visual comparisons between the original and reconstructed trajectories can be seen in Table 5.8.

5.5 Summary

In this chapter, we have introduced a global trajectory estimation method to reconstruct a holistic view of the movement of individuals in a crowd, tracked using multiple noisy sensors with incomplete coverage. Experiments using synthetic data show that our framework is robust to arbitrary amounts of noise and missing trajectory information, and is able to reconstruct complete trajectories that satisfy movement, collision, and energy constraints, while inheriting the behavioral characteristics of the original crowd.

Table 5.1: Brief description and snapshot of trajectories of simulation scenarios used to validate our framework.

bottleneck-evacuation	bottleneck-evacuation-2	bottleneck-squeeze
		
concentric-circles	hallway-two-way	hallway-four-way
		

There are many aspects of our framework that can be further improved. The framework is currently offline, as the computational complexity of our optimization approach is combinatorial with the number of individuals in a crowd. However, our method is fundamentally operable in a real-time setting, as it does not require the provision of future positions of individuals. Subsequent algorithmic improvements, including the investigation of decentralized strategies, as well as parallelization, are promising avenues of future exploration. Another aspect is the fact that our framework operates in an unsupervised way. While this is beneficial in certain circumstances, it is a promising direction to investigate the use of prior knowledge which can be used to train models for improving the efficacy of our method, both in terms of accuracy and computational complexity. In particular, we would like to use synthetic data obtained using crowd simulation methods for training models for tracking refinement methods.

Table 5.2: Root mean squared (RMS) error (in meters) of reconstructed trajectory versus ground truth. Please refer to text for more analysis on the results. For all cases, we compared the performance of the proposed framework with mean filtering as baseline. When applying the median filters for missing data, the data was preprocessed to fill in missing parts with linear interpolation. For noise removal purposes, we additionally compared with the original corrupted trajectory to demonstrate that our framework is a significant improvement compared to doing nothing in cases of non-trivial amounts of noise. Note that for missing value experiments in this table, we assumed input was noise-free thus the baseline without any filtering method was not considered.

	Noise (SNR, dB)			Missing Value (%)		
Scenario (environment width x height)	70	50	30	10	20	30
bottleneck-evacuation (200 × 160)	**0.11 ± 0.00**	**0.30 ± 0.01**	**0.93 ± 0.03**	**0.21 ± 0.04**	**0.99 ± 0.18**	3.27 ± 2.68
(median filter)	2.34 ± 0.11	2.37 ± 0.11	2.54 ± 0.10	2.34 ± 0.11	2.38 ± 0.14	**2.53 ± 0.18**
(no filtering)	0.15 ± 0.00	0.47 ± 0.01	1.48 ± 0.05			
bottleneck-evacuation-2 (100 × 80)	0.13 ± 0.00	0.15 ± 0.01	**0.30 ± 0.01**	**0.14 ± 0.00**	**0.34 ± 0.02**	**0.59 ± 0.03**
(median filter)	1.34 ± 0.03	1.35 ± 0.03	1.38 ± 0.04	1.34 ± 0.03	1.38 ± 0.03	1.48 ± 0.03
(no filtering)	**0.04 ± 0.00**	**0.14 ± 0.00**	0.45 ± 0.02			
bottleneck-squeeze (200 × 200)	**0.11 ± 0.00**	**0.27 ± 0.01**	**0.83 ± 0.01**	**0.15 ± 0.02**	**0.73 ± 0.12**	**1.78 ± 0.15**
(median filter)	2.42 ± 0.15	2.44 ± 0.15	2.57 ± 0.14	2.41 ± 0.15	2.43 ± 0.15	2.49 ± 0.17
(no filtering)	0.13 ± 0.00	0.42 ± 0.01	1.33 ± 0.03			
concentric-circles (20 × 20)	0.06 ± 0.00	0.07 ± 0.00	**0.11 ± 0.00**	**0.06 ± 0.00**	**0.16 ± 0.00**	**0.45 ± 0.00**
(median filter)	0.64 ± 0.00	0.64 ± 0.00	0.65 ± 0.00	0.63 ± 0.00	0.65 ± 0.00	0.81 ± 0.00
(no filtering)	**0.01 ± 0.00**	**0.05 ± 0.00**	0.14 ± 0.00			
hallway-two-way (200 × 200)	**0.14 ± 0.01**	**0.41 ± 0.02**	**1.28 ± 0.04**	**0.19 ± 0.01**	**0.44 ± 0.04**	**1.23 ± 0.06**
(median filter)	1.82 ± 0.12	1.87 ± 0.12	2.19 ± 0.10	1.82 ± 0.12	1.87 ± 0.11	2.09 ± 0.09
(no filtering)	0.21 ± 0.01	0.65 ± 0.02	2.06± 0.07			
hallway-four-way (200 × 200)	**0.14 ± 0.01**	**0.43 ± 0.02**	**1.32 ± 0.05**	**0.19 ± 0.01**	**0.54 ± 0.07**	**1.52 ± 0.08**
(median filter)	1.78 ± 0.11	1.83 ± 0.12	2.17 ± 0.10	1.78 ± 0.11	1.82 ± 0.11	2.04 ± 0.10
(no filtering)	0.21 ± 0.01	0.68 ± 0.02	2.13 ± 0.08			

Table 5.3: Performance of our method on simulated crowd data with noisy trajectories (SNR 50 dB). Reference, noisy trajectories are shown in black (left), and corresponding reconstructed trajectories are shown in blue (right).

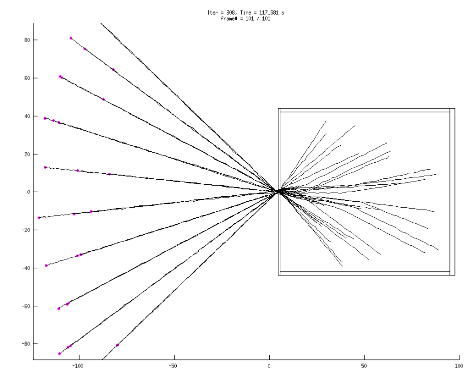
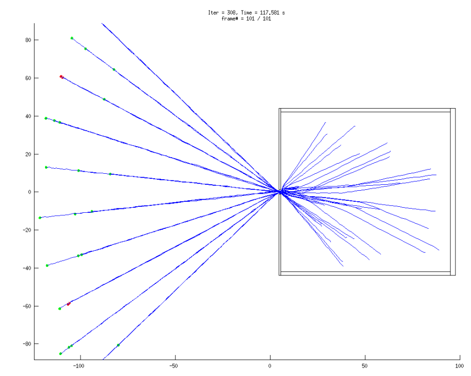
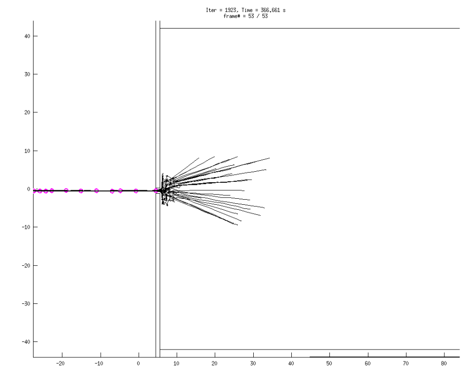
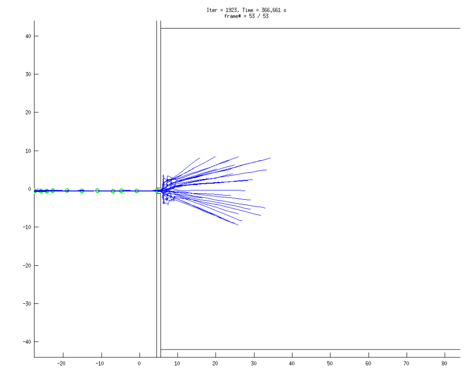
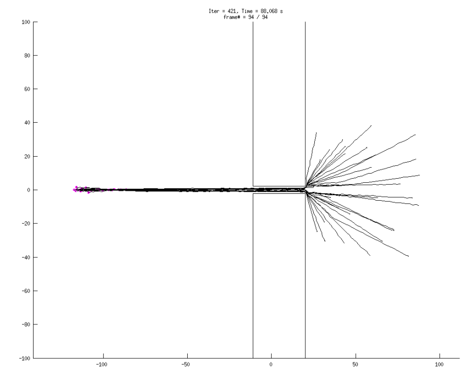
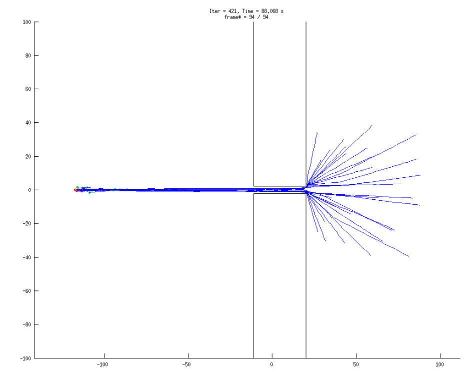
bottleneck-evacuation	
	
bottleneck-evacuation-2	
	
bottleneck-evacuation-squeeze	
	

Table 5.4: Performance of our method on simulated crowd data with noisy trajectories (SNR 50 dB). Reference, noisy trajectories are shown in black (left), and corresponding reconstructed trajectories are shown in blue (right).

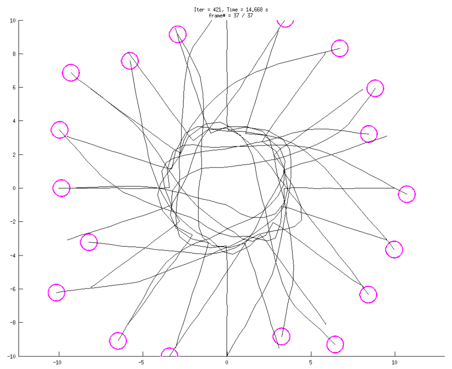
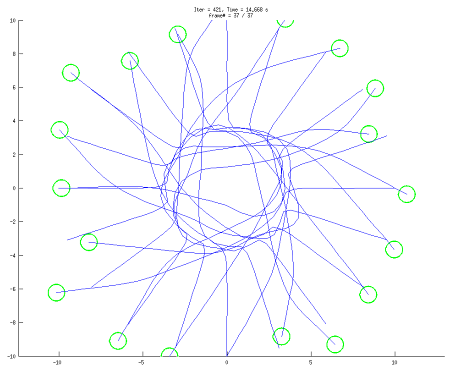
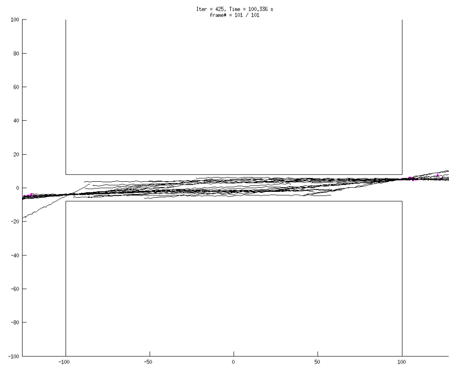
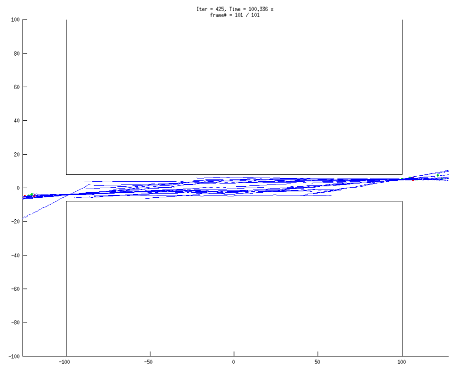
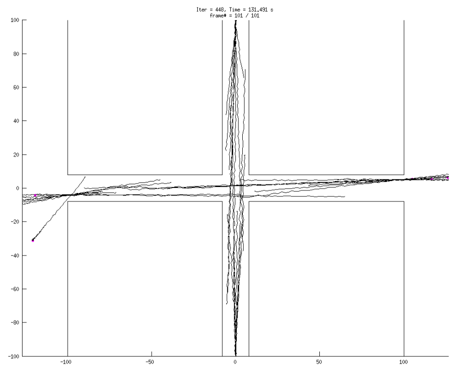
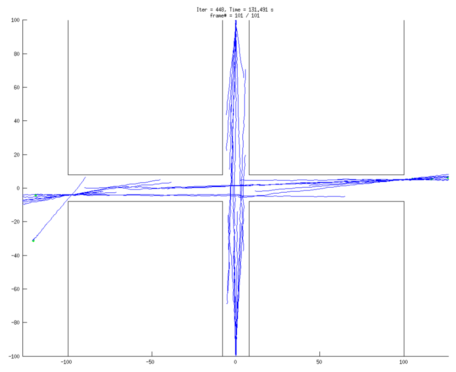
concentric-circle	
	
hallway-two-way	
	
hallway-four-way	
	

Table 5.5: Input with missing information (20% missing, missing parts marked as red, left) and reconstructed trajectories (right)

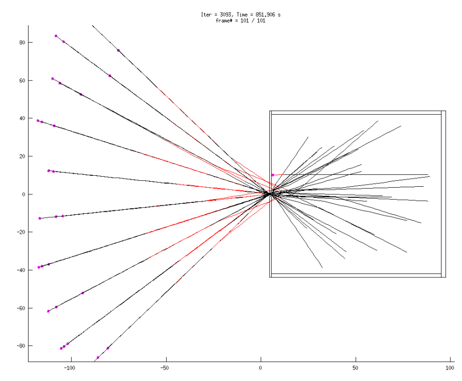
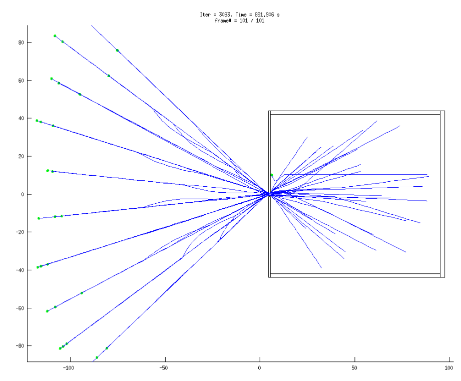
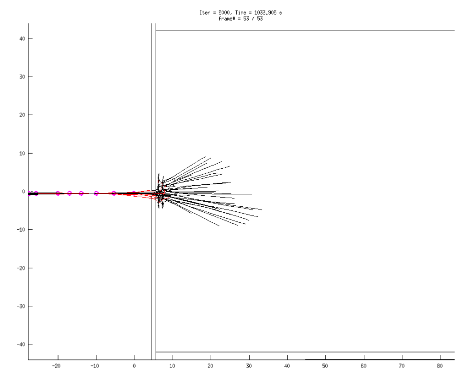
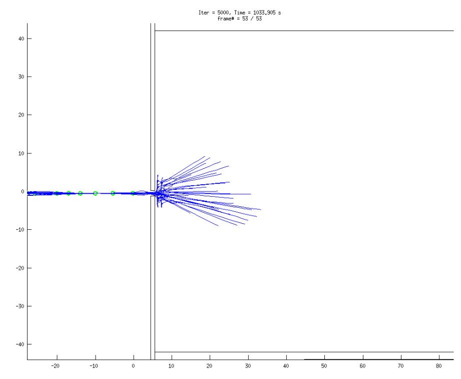
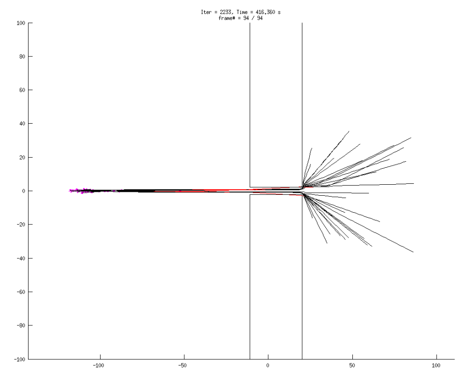
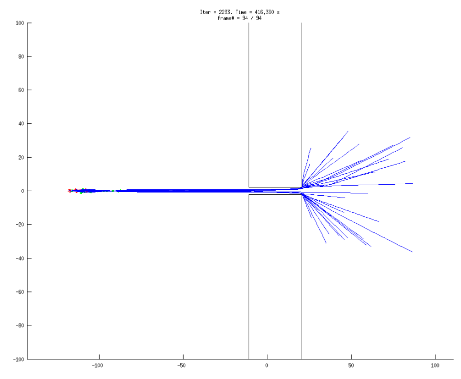
bottleneck-evacuation	
	
bottleneck-evacuation-2	
	
bottleneck-evacuation-squeeze	
	

Table 5.6: Input with missing information (20% missing, missing parts marked as red, left) and reconstructed trajectories (right)

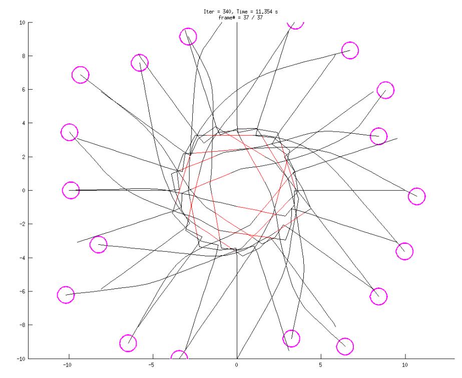
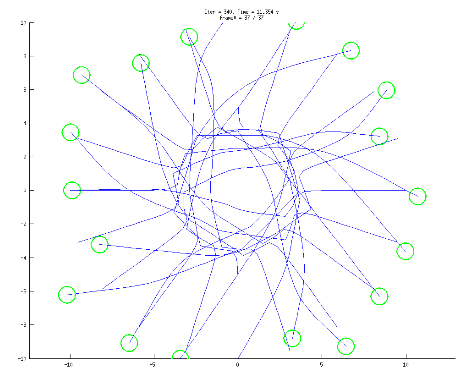
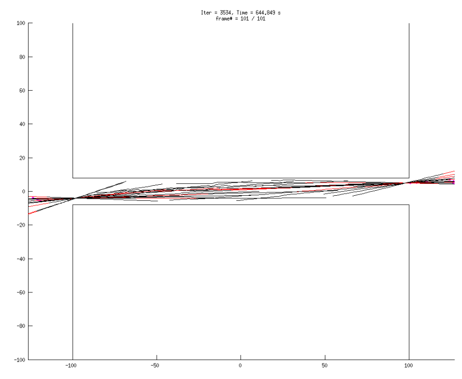
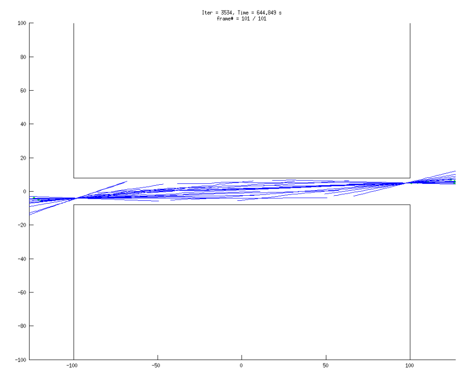
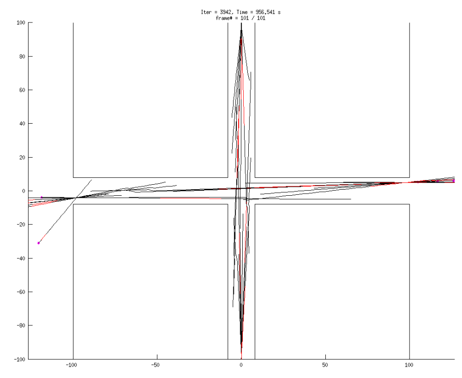
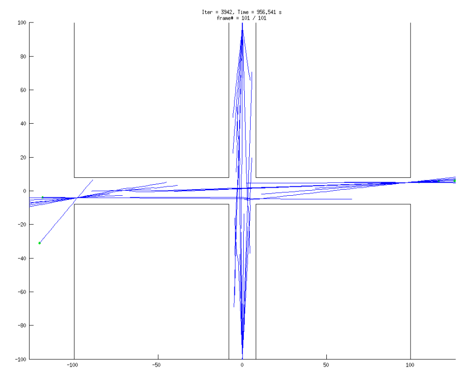
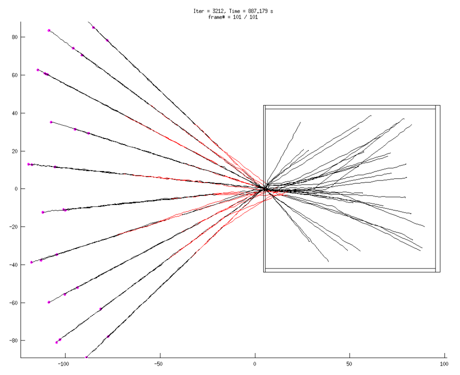
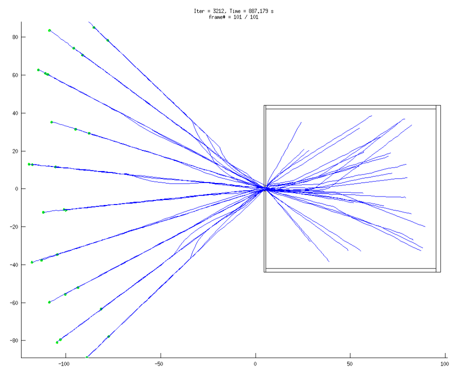
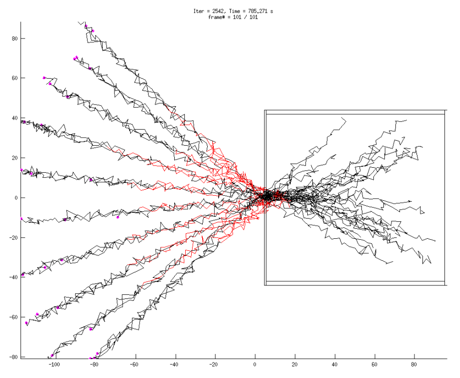
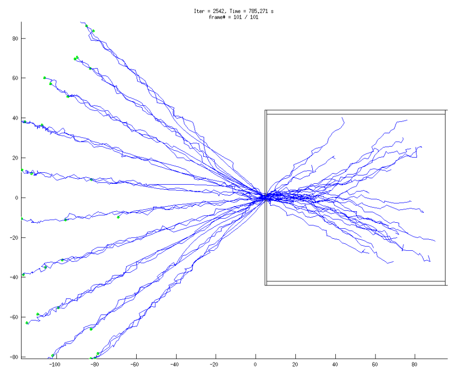
concentric-circle	
	
hallway-two-way	
	
hallway-four-way	
	

Table 5.7: Root mean squared error (in meters) of reconstructed trajectory versus ground truth when both noise and missing information exist. We show the scenario where our method suffers when compared to the baseline median filter in Table 5.2.

	Medium Noise (SNR 50 dB)		
Missing Rate (%)	10	20	30
bottleneck-evacuation	**0.24 ± 0.04**	**1.00 ± 0.18**	**2.36 ± 0.17**
(median filter)	2.35 ± 0.11	2.38 ± 0.14	2.53 ± 0.18

	Heavy Noise (SNR 30 dB)		
Missing Rate (%)	10	20	30
bottleneck-evacuation	**0.10 ± 0.03**	**1.50 ± 0.20**	2.91 ± 0.16
(median filter)	2.53 ± 0.09	2.57 ± 0.12	**2.70 ± 0.15**

Table 5.8: Noisy input with missing information (20% missing, missing parts marked as red, left) and reconstructed trajectories (right) for bottleneck-evacuation scenario.

SNR 50 dB	
	
SNR 30 dB	
	

Chapter 6

Conclusion and Future Work

In this chapter, we summarize and restate the contributions of the thesis followed by potential future research directions. First, we claim the following three major contributions:

- We proposed distributed learning models and the generalized learning framework. We showed its ability to convey strengths of probabilistic learning models in a distributed fashion, including dealing with missing values and robustness to noise. We also showed that our framework can be effectively applied to a distributed computer vision problem.
- We introduced extensions of our framework. First, we showed that this framework can be extended to naturally adapt the learnt model to dynamically changing data in an online fashion, by reformulating the framework into a Bayesian learning model. Second, we proposed a general extension of the underlying distributed optimization algorithm so that it empirically converges more quickly than the standard counterpart. We demonstrated the effectiveness of these extensions using similar sets of experiments as with the original proposed framework.
- We showed that the consensus-based optimization framework can potentially be useful in global optimization-based crowd trajectory estimation problems. Based on a prior work on a multi-agent trajectory estimation method, we showed that the framework can be used to find the optimal set of trajectories for trajectories obtained by human crowds, and it also may be useful for simulation analyses.

In summary, we proposed the general distributed learning modeling framework,

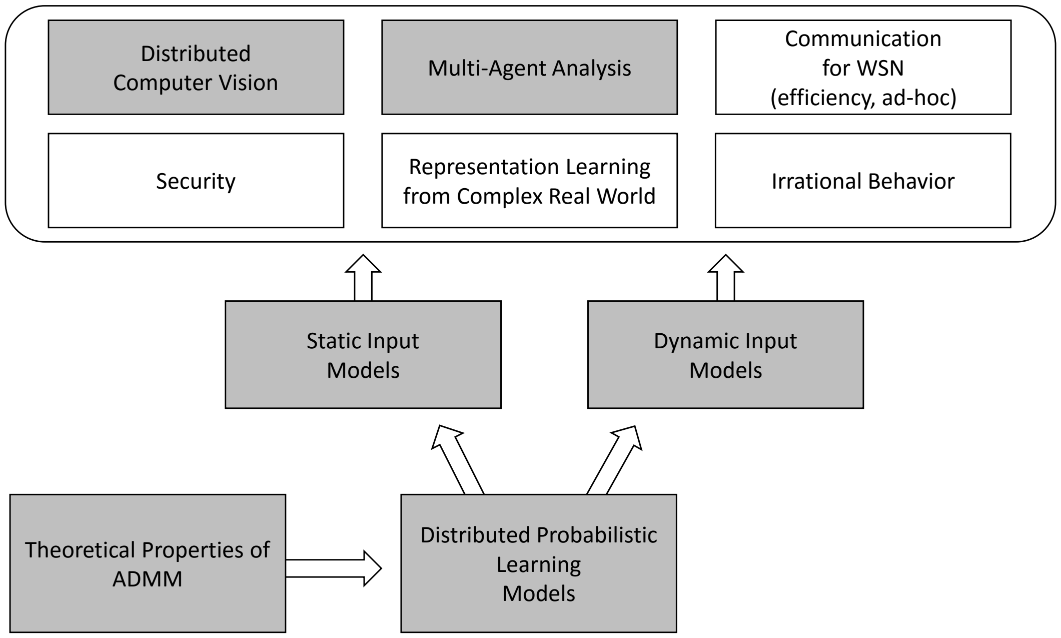

Distributed Computer Vision
Multi-Agent Analysis
Communication for WSN (efficiency, ad-hoc)
Security
Representation Learning from Complex Real World
Irrational Behavior
Static Input Models
Dynamic Input Models
Theoretical Properties of ADMM
Distributed Probabilistic Learning Models

Figure 6.1: Potential extensions and applications of proposed generalized distributed learning framework

showed examples that can handle either static or dynamic input data, extended an underlying optimization algorithm, and showed potential applications in computer vision and graphics. A pictorial description of the contributions is given in Figure 6.1. The next question is, what other applications / extensions can be done with the proposed framework. We suggest the following potential directions as a guideline:

- Throughout the thesis, we have not shed light on rather traditional issues of distributed learning in sensor networks. One important aspect of those is the communication overhead. Since the optimiztion algorithm we use is essentially iterative in nature, it requires many communications between local neighbors. This may not be a significant issue if the convergence is reasonably fast and the distance from local nodes to the processing center is very far, thus there is merit in doing multiple local communication compared to single long distance communication to the central node. Still, it is desirable to have a bounded guarantee on the number of communications needed. Another important aspect is our assumption that the network is static. Wide area networks, particularly in extreme areas, are often wireless, and are frequently connected to ad hoc neighbors. Thus, it is desirable to see how our framework can be extended to such an ad hoc environment.

- Security over the sensor network is an important concern in many cases. If our framework can provide a way to secure communication and still preserve the desired properties of decentralized learning, it would be beneficial to many applications, e.g., smart home / city for healthcare applications.

- In this thesis, and in most works in the distributed learning literature, little interest is given to the problem of finding the best representation of the distributed data before distribution. The problem is trivial in induced distributed cases, since we can always preprocess the data. However, for intrinsically distributed data, it is an important open problem that will become more and more important as the quality of the sensors improves. This is because the amount of data sensors can collect, and the communication cost, have an unavoidable trade-off relationship. In other words, the more we collect data from high quality sensors, the more information we have to send over the network if we do not know the proper representation with minimal loss of information. Therefore, (automatically) finding a good, if not the best, representation for distributed data is an important next step for distributed learning.

- Finally, it is worth noting that there is a significant gap between distributed learning and real human group learning. If one wants to utilize the proposed distributed learning framework to make a multi-agent system mimicking human groups, one has to deal with a lot of issues in terms of psychological aspects of human group behaviors. Humans can make highly varied, and sometimes irrational decisions, compared to a group of ideal individuals. While these are all interesting topics for future research, there is much work to be done prior to being able to achieve these goals.

Appendix A

Alternating Direction Method of Multipliers (ADMM)

In this appendix, we briefly review the Alternating Direction Method of Multilpiers (ADMM) and provide defintions of primal and dual residuals. More detailed explanation with examples can be found in survey [15]. Formally, we want to solve optimization problems in the form

$$\begin{aligned} \min \quad & f(\mathbf{x}) + g(\mathbf{y}) \\ s.t. \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{c}, \end{aligned} \tag{A.1}$$

where $f : \mathbb{R}^{D_x} \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^{D_y} \to \mathbb{R} \cup \{+\infty\}$ are closed, proper convex functions, $\mathbf{x} \in \mathbb{R}^{D_x}$ and $\mathbf{y} \in \mathbb{R}^{D_y}$ are variables and $\mathbf{A} \in \mathbb{R}^{D_c \times D_x}$, $\mathbf{B} \in \mathbb{R}^{D_c \times D_y}$ and $\mathbf{c} \in \mathbb{R}^{D_c}$ are known with D_x, D_y, D_c denote the dimension of the corresponding variables. As the constraint is equality constraint, we can consider the Lagrangian for A.1 as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}), \tag{A.2}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{D_c}$ is called Lagrange multiplier (or dual variable). To find optimal values for the variables, we can use the dual ascent method, i.e., iterative updates

$$(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) = \arg\min_{\mathbf{x},\mathbf{y}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}^t) \tag{A.3}$$

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \alpha^t \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t+1} - \mathbf{c}\right), \tag{A.4}$$

where $\alpha^t > 0$ denotes a step size at iteration t. The augmented Lagrangian methods provide robustness to the dual ascent method by introducing additional, squared regularization term as

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}\|_2^2, \tag{A.5}$$

where ρ is a fixed parameter that defines the amount of regularization. By applying the dual ascent to this augmented Lagrangian, we obtain iterative updates of so-called method of multipliers [23, 24] as

$$(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) = \arg\min_{\mathbf{x},\mathbf{y}} \mathcal{L}_\rho(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}^t) \tag{A.6}$$

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \rho \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t+1} - \mathbf{c}\right). \tag{A.7}$$

The ADMM algorithm has a similar form, but instead of jointly update $\mathbf{x}$ and $\mathbf{y}$, it updates one variable at a time, either sequantially or *alternatingly* as

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \mathcal{L}_\rho(\mathbf{x}, \mathbf{y}^t, \boldsymbol{\lambda}^t) \tag{A.8}$$

$$\mathbf{y}^{t+1} = \arg\min_{\mathbf{y}} \mathcal{L}_\rho(\mathbf{x}^{t+1}, \mathbf{y}, \boldsymbol{\lambda}^t) \tag{A.9}$$

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \rho \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t+1} - \mathbf{c}\right). \tag{A.10}$$

The necessary and sufficient optimality conditions for the problem (A.1) are primal feasibility

$$\mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\hat{\mathbf{y}} - \mathbf{c} = 0 \tag{A.11}$$

and dual feasibility

$$0 = \nabla f(\hat{\mathbf{x}}) + \mathbf{A}^\top \hat{\boldsymbol{\lambda}} \tag{A.12}$$

$$0 = \nabla g(\hat{\mathbf{y}}) + \mathbf{B}^\top \hat{\boldsymbol{\lambda}} \tag{A.13}$$

where $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ are feasible solutions and ∇ is the gradient of functions f, g thus here we assume f, g are differentiable. The primal residual at iteration $t + 1$ is defined as the primal feasibility, i.e.,

$$\mathbf{r}^{t+1} = \mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t+1} - \mathbf{c} \tag{A.14}$$

since if $\|\mathbf{r}^{t+1}\|_2^2 = 0$, (A.11) is satisfied. For the dual residual, we need to see what needs to be reduced in order to satisfy (A.12) and (A.13). By the iteration (A.9), $\mathbf{y}^{t+1}$

minimizes $\mathcal{L}_\rho(\mathbf{x}^{t+1}, \mathbf{y}, \boldsymbol{\lambda}^t)$. Taking derivative with respect to $\mathbf{y}^{t+1}$ on (A.5), we get

$$0 = \nabla g(\mathbf{y}^{t+1}) + \mathbf{B}^\top \boldsymbol{\lambda}^t + \rho \mathbf{B}^\top \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t+1} - \mathbf{c} \right) \tag{A.15}$$

$$= \nabla g(\mathbf{y}^{t+1}) + \mathbf{B}^\top \left\{ \boldsymbol{\lambda}^t + \rho \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t+1} - \mathbf{c} \right) \right\} \tag{A.16}$$

$$= \nabla g(\mathbf{y}^{t+1}) + \mathbf{B}^\top \boldsymbol{\lambda}^{t+1} \tag{A.17}$$

thus we see that $\mathbf{y}^{t+1}$ and $\boldsymbol{\lambda}^{t+1}$ satisfy (A.13). Similarly, $\mathbf{x}^{t+1}$ minimizes $\mathcal{L}_\rho(\mathbf{x}, \mathbf{y}^t, \boldsymbol{\lambda}^t)$ and we can see that

$$0 = \nabla f(\mathbf{x}^{t+1}) + \mathbf{A}^\top \boldsymbol{\lambda}^t + \rho \mathbf{A}^\top \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t} - \mathbf{c} \right) \tag{A.18}$$

$$= \nabla f(\mathbf{x}^{t+1}) + \mathbf{A}^\top \left\{ \boldsymbol{\lambda}^t + \rho \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t} - \mathbf{c} \right) \right\} \tag{A.19}$$

$$= \nabla f(\mathbf{x}^{t+1}) + \mathbf{A}^\top \left\{ \boldsymbol{\lambda}^t + \rho \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t+1} - \mathbf{c} \right) + \rho(\mathbf{B}\mathbf{y}^t - \mathbf{B}\mathbf{y}^{t+1}) \right\} \tag{A.20}$$

$$= \nabla f(\mathbf{x}^{t+1}) + \mathbf{A}^\top \boldsymbol{\lambda}^{t+1} + \rho \mathbf{A}^\top (\mathbf{B}\mathbf{y}^t - \mathbf{B}\mathbf{y}^{t+1}). \tag{A.21}$$

Therefore, we can define the dual residual at iteration $t+1$ as

$$\mathbf{s}^{t+1} = \rho \mathbf{A}^\top (\mathbf{B}\mathbf{y}^{t+1} - \mathbf{B}\mathbf{y}^t) \tag{A.22}$$

since $\|\mathbf{s}^{t+1}\|_2^2 = 0$ corresponds to satisfying (A.12).

Appendix B
Full Derivation of D-PPCA

In this appendix, we provide the full derivation of the proposed D-PPCA algorithm. This derivation was previously published as a part of supplementary materials for the author's conference paper [1]. First, we provide a quick reference to notations like following.

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: Undirected connected graph with vertices in $\mathcal{V}$ and edges in $\mathcal{E}$
- $i, j \in \mathcal{V}$: Node index
- $e_{ij} = (i, j) \in \mathcal{E}$: Edge connecting node i and node j
- $\mathcal{B}_i = \{j; e_{ij} \in \mathcal{E}\}$: Set of neighbor nodes directly connected to i-th node
- N_i: The number of samples collected in i-th node
- $\mathbf{z}_{in}$: n-th M dimensional latent variable at node i where $n = 1, ..., N_i$
- $\mathbf{x}_{in}$: n-th D dimensional column vector at node i where $n = 1, ..., N_i$
- $\mathbf{Z}_i = \{\mathbf{z}_{in}; n = 1, ..., N_i\}$
- $\mathbf{X}_i = \{\mathbf{x}_{in}; n = 1, ..., N_i\}$
- $\mathbf{W}_i$, $\boldsymbol{\mu}_i$, a_i: PPCA model parameters

In the distributed probabilistic model setting, we impose consensus constraints on parameters for each node. With the introduction of auxiliary variables, we can assure that all parameters reach consensus only by local optimizations. Using this idea, Forero, et al. proposed an iterative EM algorithm for the Gaussian mixture model [36]. Using a similar approach, we derive an iterative EM algorithm for PPCA. In the centralized setting, the local optimization problem using expectation on the complete data log likelihood with respect to the posterior of the latent variable is

$$\min_{\{f_{\mathbf{Z}_i},\mathbf{W}_i,\boldsymbol{\mu}_i,a_i:i\in\mathcal{V}\}} -F(f_{\mathbf{Z}_i},\mathbf{W}_i,\boldsymbol{\mu}_i,a_i) = -\mathbb{E}_{f_{\mathbf{Z}_i}}\left[\sum_{n=1}^{N_i}\log p(\mathbf{x}_{in},\mathbf{z}_{in}|\mathbf{W}_i,\boldsymbol{\mu}_i,a_i^{-1})\right]$$

where $f_{\mathbf{Z}_i} = p(\mathbf{Z}_i|\mathbf{X}_i)$[1]. If we impose the consensus constraints on this, the constrained local optimization problem becomes

$$\begin{aligned}
\min_{\{f_{\mathbf{Z}_i},\mathbf{W}_i,\boldsymbol{\mu}_i,a_i:i\in\mathcal{V}\}} & -F(f_{\mathbf{Z}_i},\mathbf{W}_i,\boldsymbol{\mu}_i,a_i)\\
s.t. \quad & \mathbf{W}_i=\boldsymbol{\rho}_{ij},\quad \boldsymbol{\rho}_{ij}=\mathbf{W}_j \quad i\in\mathcal{V}, j\in\mathcal{B}_i\\
& \boldsymbol{\mu}_i=\boldsymbol{\phi}_{ij},\quad \boldsymbol{\phi}_{ij}=\boldsymbol{\mu}_j \quad i\in\mathcal{V}, j\in\mathcal{B}_i\\
& a_i=\psi_{ij},\quad \psi_{ij}=a_j \quad i\in\mathcal{V}, j\in\mathcal{B}_i
\end{aligned} \tag{B.1}$$

where $\boldsymbol{\rho}_{ij},\boldsymbol{\phi}_{ij},\psi_{ij}$ are auxiliary variables. If we solve this local optimization problem, we also solve global optimization since global optimization is simply the summation of local ones given consensus constraints meet. The augmented Lagrangian of (B.1) is

$$\begin{aligned}
\mathcal{L}(\Phi_i) = & -F(f_{\mathbf{Z}_i},\mathbf{W}_i,\boldsymbol{\mu}_i,a_i)\\
& +\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}\left(\boldsymbol{\lambda}_{ij1}^{\top}(\mathbf{W}_i-\boldsymbol{\rho}_{ij})+\boldsymbol{\lambda}_{ij2}^{\top}(\boldsymbol{\rho}_{ij}-\mathbf{W}_j)\right)\\
& +\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}\left(\boldsymbol{\gamma}_{ij1}^{\top}(\boldsymbol{\mu}_i-\boldsymbol{\phi}_{ij})+\boldsymbol{\gamma}_{ij2}^{\top}(\boldsymbol{\phi}_{ij}-\boldsymbol{\mu}_j)\right)\\
& +\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}(\beta_{ij1}(a_i-\psi_{ij})+\beta_{ij2}(\psi_{ij}-a_j))\\
& +\frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}(||\mathbf{W}_i-\boldsymbol{\rho}_{ij}||^2+||\boldsymbol{\rho}_{ij}-\mathbf{W}_j||^2)\\
& +\frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}(||\boldsymbol{\mu}_i-\boldsymbol{\phi}_{ij}||^2+||\boldsymbol{\phi}_{ij}-\boldsymbol{\mu}_j||^2)\\
& +\frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}((a_i-\psi_{ij})^2+(\psi_{ij}-a_j)^2)
\end{aligned} \tag{B.2}$$

where $\Phi_i = \{f_{\mathbf{Z}_i},\mathbf{W}_i,\boldsymbol{\mu}_i,a_i,\boldsymbol{\rho}_{ij},\boldsymbol{\phi}_{ij},\psi_{ij}; i\in\mathcal{V}, j\in\mathcal{B}_i\}$ and $\{\boldsymbol{\lambda}_{ijk}\},\{\boldsymbol{\gamma}_{ijk}\},\{\beta_{ijk}\}$ with $k=1,2$ are the Lagrange multipliers. η is a positive scalar and $||\cdot||$ denotes the induced

[1]In main chapter, we defined the optimization using marginal distribution to make it consistent with our general distributed probabilistic model. However, one can optimize the expected value of the completed log likelihood with respect to posterior as shown here.

norm. We cyclically minimize $\mathcal{L}(\Phi_i)$ over its parameters, then follow a gradient ascent step over the Lagrange multipliers. The iterates, using t as iteration index, are

$$f_{\mathbf{Z}}^{(t+1)} = \arg\min_{f_{\mathbf{Z}}} \mathcal{L}(f_{\mathbf{Z}}, \mathbf{W}_i^{(t)}, \boldsymbol{\mu}_i^{(t)}, a_i^{(t)}, \boldsymbol{\rho}_{ij}^{(t)}, \boldsymbol{\phi}_{ij}^{(t)}, \psi_{ij}^{(t)}), \tag{B.3}$$

$$\mathbf{W}_i^{(t+1)} = \arg\min_{\mathbf{W}_i} \mathcal{L}(f_{\mathbf{Z}}^{(t+1)}, \mathbf{W}_i, \boldsymbol{\mu}_i^{(t)}, a_i^{(t)}, \boldsymbol{\rho}_{ij}^{(t)}, \boldsymbol{\phi}_{ij}^{(t)}, \psi_{ij}^{(t)}), \tag{B.4}$$

$$\boldsymbol{\mu}_i^{(t+1)} = \arg\min_{\boldsymbol{\mu}_i} \mathcal{L}(f_{\mathbf{Z}}^{(t+1)}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i, a_i^{(t)}, \boldsymbol{\rho}_{ij}^{(t)}, \boldsymbol{\phi}_{ij}^{(t)}, \psi_{ij}^{(t)}), \tag{B.5}$$

$$a_i^{(t+1)} = \arg\min_{a_i} \mathcal{L}(f_{\mathbf{Z}}^{(t+1)}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i, \boldsymbol{\rho}_{ij}^{(t)}, \boldsymbol{\phi}_{ij}^{(t)}, \psi_{ij}^{(t)}), \tag{B.6}$$

$$\boldsymbol{\rho}_{ij}^{(t+1)} = \arg\min_{\boldsymbol{\rho}_{ij}} \mathcal{L}(f_{\mathbf{Z}}^{(t+1)}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i^{(t+1)}, \boldsymbol{\rho}_{ij}, \boldsymbol{\phi}_{ij}^{(t)}, \psi_{ij}^{(t)}), \tag{B.7}$$

$$\boldsymbol{\phi}_{ij}^{(t+1)} = \arg\min_{\boldsymbol{\phi}_{ij}} \mathcal{L}(f_{\mathbf{Z}}^{(t+1)}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i^{(t+1)}, \boldsymbol{\rho}_{ij}^{(t+1)}, \boldsymbol{\phi}_{ij}, \psi_{ij}^{(t)}), \tag{B.8}$$

$$\psi_{ij}^{(t+1)} = \arg\min_{\psi_{ij}} \mathcal{L}(f_{\mathbf{Z}}^{(t+1)}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i^{(t+1)}, \boldsymbol{\rho}_{ij}^{(t+1)}, \boldsymbol{\phi}_{ij}^{(t+1)}, \psi_{ij}), \tag{B.9}$$

$$\boldsymbol{\lambda}_{ij1}^{(t+1)} = \boldsymbol{\lambda}_{ij1}^{(t)} + \eta \left[\mathbf{W}_i^{(t+1)} - \boldsymbol{\rho}_{ij}^{(t+1)}\right], \forall i \in \mathcal{V}, j \in \mathcal{B}_i, \tag{B.10}$$

$$\boldsymbol{\lambda}_{ij2}^{(t+1)} = \boldsymbol{\lambda}_{ij2}^{(t)} + \eta \left[\boldsymbol{\rho}_{ij}^{(t+1)} - \mathbf{W}_j^{(t+1)}\right], \forall i \in \mathcal{V}, j \in \mathcal{B}_i, \tag{B.11}$$

$$\boldsymbol{\gamma}_{ij1}^{(t+1)} = \boldsymbol{\gamma}_{ij1}^{(t)} + \eta \left[\boldsymbol{\mu}_i^{(t+1)} - \boldsymbol{\phi}_{ij}^{(t+1)}\right], \forall i \in \mathcal{V}, j \in \mathcal{B}_i, \tag{B.12}$$

$$\boldsymbol{\gamma}_{ij2}^{(t+1)} = \boldsymbol{\gamma}_{ij2}^{(t)} + \eta \left[\boldsymbol{\phi}_{ij}^{(t+1)} - \boldsymbol{\mu}_j^{(t+1)}\right], \forall i \in \mathcal{V}, j \in \mathcal{B}_i, \tag{B.13}$$

$$\beta_{ij1}^{(t+1)} = \beta_{ij1}^{(t)} + \eta \left[a_i^{(t+1)} - \psi_{ij}^{(t+1)}\right], \forall i \in \mathcal{V}, j \in \mathcal{B}_i, \tag{B.14}$$

$$\beta_{ij2}^{(t+1)} = \beta_{ij2}^{(t)} + \eta \left[\psi_{ij}^{(t+1)} - a_j^{(t+1)}\right], \forall i \in \mathcal{V}, j \in \mathcal{B}_i. \tag{B.15}$$

Computing (B.3): Omitting t notations, the first term of (B.2)

$$\begin{aligned} -F(f_{\mathbf{Z}_i}, \mathbf{W}_i, \boldsymbol{\mu}_i, a_i) = \sum_{n=1}^{N_i} \Bigg\{ & \frac{M}{2}\log(2\pi) + \frac{1}{2} tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]\right] - \frac{D}{2}\log(2\pi a_i) \\ & + \frac{a_i}{2}||\mathbf{x}_{in} - \boldsymbol{\mu}_i||^2 + \frac{a_i}{2} tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]\mathbf{W}_i^\top \mathbf{W}_i\right] \\ & - a_i \mathbb{E}[\mathbf{z}_{in}]^\top \mathbf{W}_i^\top (\mathbf{x}_{in} - \boldsymbol{\mu}_i) \Bigg\} \end{aligned} \tag{B.16}$$

is the only term dependent on $f_{\mathbf{Z}_i}$. Thus, if all other parameters and multipliers are fixed at the t-th iteration, (B.3) can be computed using the expected values of the latent variables as we did in the E-step of the centralized setting. By using the posterior distribution of the centralized PPCA, we compute

$$\mathbb{E}[\mathbf{z}_{in}] = \mathbf{L}_i^{-1}\mathbf{W}_i^\top(\mathbf{x}_{in} - \boldsymbol{\mu}_i) \tag{B.17}$$

$$\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top] = a_i^{-1}\mathbf{L}_i^{-1} + \mathbb{E}[\mathbf{z}_{in}]\mathbb{E}[\mathbf{z}_{in}]^\top \tag{B.18}$$

and we plug (B.17) and (B.18) into (B.16).

Computing (B.7)-(B.15): Auxiliary variables, $\boldsymbol{\rho}_{ij}, \boldsymbol{\phi}_{ij}, \psi_{ij}$ are independent from (B.16). Thus (B.7)-(B.9) are linear-quadratic optimization with respect to these variables and we can find a closed-form solution for these variables. For (B.9), this yields

$$\begin{aligned}\frac{\partial \mathcal{L}(\Phi_i)}{\partial \psi_{ij}} = \frac{\partial}{\partial \psi_{ij}}\Bigg\{ & \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}\left(\beta_{ij1}^{(t)}(a_i^{(t+1)} - \psi_{ij}) + \beta_{ij2}^{(t)}(\psi_{ij} - a_j^{(t+1)})\right) \\ & + \frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}((a_i^{(t+1)} - \psi_{ij})^2 + (\psi_{ij} - a_j^{(t+1)})^2)\Bigg\} \\ 0 = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}\Bigg\{ & -\left(\beta_{ij1}^{(t)} - \beta_{ij2}^{(t)}\right) - \eta\left(a_i^{(t+1)} + a_j^{(t+1)}\right) + 2\eta\psi_{ij}\Bigg\}\end{aligned}$$

Since the Lagrange multipliers, parameters and η are all zero or positive values, we get

$$\psi_{ij}^{(t+1)} = \frac{1}{2\eta}\left(\beta_{ij1}^{(t)} - \beta_{ij2}^{(t)}\right) + \frac{1}{2}\left(a_i^{(t+1)} + a_j^{(t+1)}\right) \tag{B.19}$$

Using the same technique on (B.7) and (B.8), we get

$$\boldsymbol{\rho}_{ij}^{(t+1)} = \frac{1}{2\eta}\left(\boldsymbol{\lambda}_{ij1}^{(t)} - \boldsymbol{\lambda}_{ij2}^{(t)}\right) + \frac{1}{2}\left(\mathbf{W}_i^{(t+1)} + \mathbf{W}_j^{(t+1)}\right), \tag{B.20}$$

$$\boldsymbol{\phi}_{ij}^{(t+1)} = \frac{1}{2\eta}\left(\boldsymbol{\gamma}_{ij1}^{(t)} - \boldsymbol{\gamma}_{ij2}^{(t)}\right) + \frac{1}{2}\left(\boldsymbol{\mu}_i^{(t+1)} + \boldsymbol{\mu}_j^{(t+1)}\right) \tag{B.21}$$

respectively. Plugging (B.20) into (B.10) and (B.11), (B.21) into (B.12) and (B.13), (B.19) into (B.14) and (B.15), we get

$$\boldsymbol{\lambda}_{ij1}^{(t+1)} = \frac{1}{2}\left(\boldsymbol{\lambda}_{ij1}^{(t)} + \boldsymbol{\lambda}_{ij2}^{(t)}\right) + \frac{\eta}{2}\left(\mathbf{W}_i^{(t+1)} - \mathbf{W}_j^{(t+1)}\right), \tag{B.22}$$

$$\boldsymbol{\lambda}_{ij2}^{(t+1)} = \frac{1}{2}\left(\boldsymbol{\lambda}_{ij1}^{(t)} + \boldsymbol{\lambda}_{ij2}^{(t)}\right) + \frac{\eta}{2}\left(\mathbf{W}_i^{(t+1)} - \mathbf{W}_j^{(t+1)}\right), \tag{B.23}$$

$$\boldsymbol{\gamma}_{ij1}^{(t+1)} = \frac{1}{2}\left(\boldsymbol{\gamma}_{ij1}^{(t)} + \boldsymbol{\gamma}_{ij2}^{(t)}\right) + \frac{\eta}{2}\left(\boldsymbol{\mu}_i^{(t+1)} - \boldsymbol{\mu}_j^{(t+1)}\right), \tag{B.24}$$

$$\boldsymbol{\gamma}_{ij2}^{(t+1)} = \frac{1}{2}\left(\boldsymbol{\gamma}_{ij1}^{(t)} + \boldsymbol{\gamma}_{ij2}^{(t)}\right) + \frac{\eta}{2}\left(\boldsymbol{\mu}_i^{(t+1)} - \boldsymbol{\mu}_j^{(t+1)}\right), \tag{B.25}$$

$$\beta_{ij1}^{(t+1)} = \frac{1}{2}\left(\beta_{ij1}^{(t)} + \beta_{ij2}^{(t)}\right) + \frac{\eta}{2}\left(a_i^{(t+1)} - a_j^{(t+1)}\right), \tag{B.26}$$

$$\beta_{ij2}^{(t+1)} = \frac{1}{2}\left(\beta_{ij1}^{(t)} + \beta_{ij2}^{(t)}\right) + \frac{\eta}{2}\left(a_i^{(t+1)} - a_j^{(t+1)}\right) \tag{B.27}$$

where $\forall i \in \mathcal{V}$ and $j \in \mathcal{B}_i$. As in Appendix B of [36], we observe that at iteration t, $\boldsymbol{\lambda}_{ij1}^{(t+1)} = \boldsymbol{\lambda}_{ij2}^{(t+1)}, \boldsymbol{\gamma}_{ij1}^{(t+1)} = \boldsymbol{\gamma}_{ij2}^{(t+1)}, \beta_{ij1}^{(t+1)} = \beta_{ij2}^{(t+1)}$ if we assume initial value of each Lagrange multiplier was set to zero. Thus, it suffices to find only one of the two. We define this one value as $\boldsymbol{\lambda}_{ij}^{(t)} := \boldsymbol{\lambda}_{ij1}^{(t)} = \boldsymbol{\lambda}_{ij2}^{(t)}$, $\boldsymbol{\gamma}_{ij}^{(t)} := \boldsymbol{\gamma}_{ij1}^{(t)} = \boldsymbol{\gamma}_{ij2}^{(t)}$ and $\beta_{ij}^{(t)} := \beta_{ij1}^{(t)} = \beta_{ij2}^{(t)}$. Moreover, if we define

$$\boldsymbol{\lambda}_i^{(t)} := \sum_{j \in \mathcal{B}_i} \boldsymbol{\lambda}_{ij}^{(t)}, \quad \boldsymbol{\gamma}_i^{(t)} := \sum_{j \in \mathcal{B}_i} \boldsymbol{\gamma}_{ij}^{(t)}, \quad \beta_i^{(t)} := \sum_{j \in \mathcal{B}_i} \beta_{ij}^{(t)},$$

then (B.22)-(B.27) reduce to

$$\boldsymbol{\lambda}_i^{(t+1)} = \boldsymbol{\lambda}_i^{(t)} + \frac{\eta}{2} \sum_{j \in \mathcal{B}_i} \left\{ \mathbf{W}_i^{(t+1)} - \mathbf{W}_j^{(t+1)} \right\}, \tag{B.28}$$

$$\boldsymbol{\gamma}_i^{(t+1)} = \boldsymbol{\gamma}_i^{(t)} + \frac{\eta}{2} \sum_{j \in \mathcal{B}_i} \left\{ \boldsymbol{\mu}_i^{(t+1)} - \boldsymbol{\mu}_j^{(t+1)} \right\}, \tag{B.29}$$

$$\beta_i^{(t+1)} = \beta_i^{(t)} + \frac{\eta}{2} \sum_{j \in \mathcal{B}_i} \left\{ a_i^{(t+1)} - a_j^{(t+1)} \right\}. \tag{B.30}$$

Computing (B.6): We tackle a_i first. Omitting t temporarily for notational brevity, the derivate of $\mathcal{L}(\boldsymbol{\Phi}_i)$ with respect to a_i is

$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\Phi}_i)}{\partial a_i} &= \frac{\partial}{\partial a_i} \Bigg[\sum_{n=1}^{N_i} \Bigg\{ -\frac{D}{2}\log(2\pi a_i) + \frac{a_i}{2}||\mathbf{x}_{in} - \boldsymbol{\mu}_i||^2 + \frac{a_i}{2} tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top] \mathbf{W}_i^\top \mathbf{W}_i \right] \\
&\qquad - a_i \mathbb{E}[\mathbf{z}_{in}]^\top \mathbf{W}_i^\top (\mathbf{x}_{in} - \boldsymbol{\mu}_i) \Bigg\} \\
&\qquad + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{B}_i} (\beta_{ij1}(a_i - \psi_{ij}) + \beta_{ij2}(\psi_{ij} - a_j)) \\
&\qquad + \frac{\eta}{2} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{B}_i} ((a_i - \psi_{ij})^2 + (\psi_{ij} - a_j)^2) \Bigg] \\
\frac{\partial \mathcal{L}(\boldsymbol{\Phi}_i)}{\partial a_i} &= \sum_{n=1}^{N_i} \Bigg\{ -\frac{D}{2} a_i^{-1} - \mathbb{E}[\mathbf{z}_{in}]^\top \mathbf{W}_i^\top (\mathbf{x}_{in} - \boldsymbol{\mu}_i) \\
&\qquad + \frac{1}{2} \left\{ ||\mathbf{x}_{in} - \boldsymbol{\mu}_i||^2 + tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top] \mathbf{W}_i^\top \mathbf{W}_i \right] \right\} \Bigg\} \\
&\qquad + \sum_{j \in \mathcal{B}_i} (\beta_{ij1} - \beta_{ji2}) + \frac{\partial}{\partial a_i} \left[\eta \sum_{j \in B_i} \left\{ (a_i - \psi_{ij})^2 + (\psi_{ji} - a_i)^2 \right\} \right] \\
&= \mathcal{Q}_{a_i}(f_{\mathbf{z}}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i) \\
&\qquad + \sum_{j \in \mathcal{B}_i} (\beta_{ij1} - \beta_{ji2}) + 2\eta \sum_{j \in B_i} (a_i - \psi_{ij}) \quad (\because \psi_{ij} = \psi_{ji})
\end{aligned} \tag{B.31}$$

where we temporarily suppressed the first summation term as $\mathcal{Q}_{a_i}(f_{\mathbf{z}}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i)$ for clearer presentation. Putting t back while plugging (B.19), the closed form solution of ψ_{ij}, into (B.31), we get

$$\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{\Phi}_i)}{\partial a_i} &= \mathcal{Q}_{a_i}(f_{\mathbf{z}}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i) + \sum_{j \in \mathcal{B}_i} \left(\beta_{ij1}^{(t)} - \beta_{ji2}^{(t)}\right) \\ &\quad + 2\eta \sum_{j \in B_i} \left(a_i - \left\{\frac{1}{2\eta}\left(\beta_{ij1}^{(t-1)} - \beta_{ij2}^{(t-1)}\right) + \frac{1}{2}\left(a_i^{(t)} + a_j^{(t)}\right)\right\}\right) \\ &= \mathcal{Q}_{a_i}(f_{\mathbf{z}}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i) + \sum_{j \in \mathcal{B}_i} \left(\beta_{ij1}^{(t)} - \beta_{ji2}^{(t)}\right) \\ &\quad + 2\eta a_i^2 |\mathcal{B}_i| - \eta \sum_{j \in B_i} \left(a_i^{(t)} + a_j^{(t)}\right) \end{aligned} \tag{B.32}$$

since $\beta_{ij1}^{(t-1)} = \beta_{ij2}^{(t-1)}$ if we assume their initial values are zeros. Here again, we can further simplify (B.32) using the following property.

Proposition 1. *(Forero, et al. [36]) Given $\beta_{ij}^{(0)} = 0$, $\beta_{ij}^{(t)} = -\beta_{ji}^{(t)}$.*

Proof. We have observed that $\beta_{ij1}^{(0)} = \beta_{ij2}^{(0)} = 0 \implies \beta_{ij1}^{(t)} = \beta_{ij2}^{(t)}$ by applying induction on (B.26) and (B.27). Thus, if we define $\beta_{ij}^{(t)} := \beta_{ij1}^{(t)} = \beta_{ij2}^{(t)}$,

$$\begin{aligned}\beta_{ij1}^{(t+1)} &= \frac{1}{2}(\beta_{ij1}^{(t)} + \beta_{ij2}^{(t)}) + \frac{\eta}{2}(a_i^{(t+1)} - a_j^{(t+1)}) = \frac{1}{2}(\beta_{ij1}^{(t)} + \beta_{ij1}^{(t)}) + \frac{\eta}{2}(a_i^{(t+1)} - a_j^{(t+1)}) \\ &= \beta_{ij1}^{(t)} + \frac{\eta}{2}(a_i^{(t+1)} - a_j^{(t+1)}) \\ \therefore \beta_{ij}^{(t+1)} &= \beta_{ij}^{(t)} + \frac{\eta}{2}(a_i^{(t+1)} - a_j^{(t+1)}).\end{aligned}$$

Therefore, given $\beta_{ij}^{(0)} = \beta_{ji}^{(0)} = 0$, $\beta_{ij}^{(1)} = -\beta_{ji}^{(1)}$. If we continue iterations,

$$\begin{aligned}\beta_{ij}^{(2)} &= \beta_{ij}^{(1)} + \frac{\eta}{2}(a_i^{(t+1)} - a_j^{(t+1)}) \\ \beta_{ji}^{(2)} &= \beta_{ji}^{(1)} + \frac{\eta}{2}(a_j^{(t+1)} - a_i^{(t+1)}) = -\beta_{ij}^{(1)} - \frac{\eta}{2}(a_i^{(t+1)} - a_j^{(t+1)}) \\ &= -\beta_{ij}^{(2)}.\end{aligned}$$

By induction, $\beta_{ij}^{(t)} = -\beta_{ji}^{(t)}$. □

Therefore, if we define $\beta_i^{(t)} = \sum_{j \in \mathcal{B}_i} \beta_{ij}^{(t)}$, (B.32) becomes

$$\frac{\partial \mathcal{L}(\boldsymbol{\Phi}_i)}{\partial a_i} = \mathcal{Q}_{a_i}(f_{\mathbf{z}}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i) + 2\beta_i^{(t)} + 2\eta a_i |\mathcal{B}_i| - \eta \sum_{j \in B_i} \left(a_i^{(t)} + a_j^{(t)}\right)$$

Setting this to zero while substituting $\mathcal{Q}_{a_i}(f_{\mathbf{z}}, \mathbf{W}_i^{(t+1)}, \boldsymbol{\mu}_i^{(t+1)}, a_i)$ back into its original form and simplifying for a_i, we get

$$\begin{aligned}
0 = &\sum_{n=1}^{N_i} \Bigg\{ -\frac{D}{2}(a_i)^{-1} - \mathbb{E}[\mathbf{z}_{in}]^\top \mathbf{W}_i^\top (\mathbf{x}_{in} - \boldsymbol{\mu}_i) \\
&\qquad + \frac{1}{2}\left\{ ||\mathbf{x}_{in} - \boldsymbol{\mu}_i||^2 + tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]\mathbf{W}_i^\top \mathbf{W}_i\right]\right\} \Bigg\} \\
&\quad + 2\beta_i^{(t)} + 2\eta a_i |\mathcal{B}_i| - \eta \sum_{j \in B_i} \left(a_i^{(t)} + a_j^{(t)}\right) \\
= &-\frac{N_i D}{2}(a_i)^{-1} - \sum_{n=1}^{N_i} \mathbb{E}[\mathbf{z}_{in}]^\top \mathbf{W}_i^\top (\mathbf{x}_{in} - \boldsymbol{\mu}_i) + 2\beta_i^{(t)} + 2\eta a_i|\mathcal{B}_i| - \eta \sum_{j\in B_i}\left(a_i^{(t)} + a_j^{(t)}\right) \\
&\quad + \frac{1}{2}\sum_{n=1}^{N_i}\left\{ ||\mathbf{x}_{in} - \boldsymbol{\mu}_i||^2 + tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]\mathbf{W}_i^\top \mathbf{W}_i\right]\right\} \\
= &-\frac{N_i D}{2} + 2\eta|\mathcal{B}_i|(a_i)^2 \\
&\quad + a_i \Bigg\{ 2\beta_i^{(t)} - \eta \sum_{j\in B_i}\left(a_i^{(t)} + a_j^{(t)}\right) - \sum_{n=1}^{N_i}\mathbb{E}[\mathbf{z}_{in}]^\top \mathbf{W}_i^\top(\mathbf{x}_{in} - \boldsymbol{\mu}_i) \\
&\qquad + \frac{1}{2}\sum_{n=1}^{N_i}\left\{ ||\mathbf{x}_{in} - \boldsymbol{\mu}_i||^2 + tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]\mathbf{W}_i^\top \mathbf{W}_i\right]\right\} \Bigg\}.
\end{aligned} \tag{B.33}$$

as we omitted t in $\mathbf{z}_{in}^{(t+1)}$, $\mathbf{W}_i^{(t+1)}$ and $\boldsymbol{\mu}_i^{(t+1)}$ for notational brevity. This is a quadratic function of a_i for which we can find an algebraic solution.

Computing (B.5): We cannot simply use the closed form solution of the centralized PPCA, i.e., $\boldsymbol{\mu} = \bar{\boldsymbol{\mu}}$. We could use this result in the centralized setting but we cannot use it in the distributed model due to the auxiliary variable constraints. Again, we omit t temporarily for brevity.

$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\Phi}_i)}{\partial \boldsymbol{\mu}_i} = \frac{\partial}{\partial \boldsymbol{\mu}_i}\Bigg[&\sum_{n=1}^{N_i}\Bigg\{ -\frac{D}{2}\log(2\pi a_i) + \frac{a_i}{2}||\mathbf{x}_{in} - \boldsymbol{\mu}_i||^2 + \frac{a_i}{2} tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]\mathbf{W}_i^\top \mathbf{W}_i\right] \\
&\qquad - a_i \mathbb{E}[\mathbf{z}_{in}]^\top \mathbf{W}_i^\top (\mathbf{x}_{in} - \boldsymbol{\mu}_i)\Bigg\} \\
&+ \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}\left(\boldsymbol{\gamma}_{ij1}^\top(\boldsymbol{\mu}_i - \boldsymbol{\phi}_{ij}) + \boldsymbol{\gamma}_{ij2}^\top(\boldsymbol{\phi}_{ij} - \boldsymbol{\mu}_j)\right) \\
&+ \frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}\left(||\boldsymbol{\mu}_i - \boldsymbol{\phi}_{ij}||^2 + ||\boldsymbol{\phi}_{ij} - \boldsymbol{\mu}_j||^2\right)\Bigg]
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{n=1}^{N_i} \left\{ -a_i(\mathbf{x}_{in} - \boldsymbol{\mu}_i) + a_i \mathbf{W}_i \mathbb{E}[\mathbf{z}_{in}] \right\} \\
&\quad + \sum_{j\in\mathcal{B}_i} (\boldsymbol{\gamma}_{ij1} - \boldsymbol{\gamma}_{ji2}) + \frac{\partial}{\partial \boldsymbol{\mu}_i} \left[\eta \sum_{j\in B_i} \left\{ ||\boldsymbol{\mu}_i - \boldsymbol{\phi}_{ij}||^2 + ||\boldsymbol{\phi}_{ji} - \boldsymbol{\mu}_i||^2 \right\} \right] \\
&= -a_i \sum_{n=1}^{N_i} \mathbf{x}_{in} + N_i a_i \boldsymbol{\mu}_i + a_i \sum_{n=1}^{N_i} \mathbf{W}_i \mathbb{E}[\mathbf{z}_{in}] \\
&\quad + \sum_{j\in\mathcal{B}_i} (\boldsymbol{\gamma}_{ij1} - \boldsymbol{\gamma}_{ji2}) + 2\eta \sum_{j\in B_i} (\boldsymbol{\mu}_i - \boldsymbol{\phi}_{ij}) \qquad \text{(B.34)}
\end{aligned}
$$

If we substitute the closed form solution of $\boldsymbol{\phi}_{ij}$, i.e., (B.21) into (B.34) while taking t notations back, we get

$$
\begin{aligned}
0 &= -a_i \sum_{n=1}^{N_i} \mathbf{x}_{in} + N_i a_i \boldsymbol{\mu}_i + a_i \sum_{n=1}^{N_i} \mathbf{W}_i \mathbb{E}[\mathbf{z}_{in}] + \sum_{j\in\mathcal{B}_i} (\boldsymbol{\gamma}_{ij1} - \boldsymbol{\gamma}_{ji2}) \\
&\quad + 2\eta \sum_{j\in B_i} \left(\boldsymbol{\mu}_i - \left\{ \frac{1}{2\eta}(\boldsymbol{\gamma}_{ij1}(t-1) - \boldsymbol{\gamma}_{ij2}(t-1)) + \frac{1}{2}(\boldsymbol{\mu}_i^{(t)} + \boldsymbol{\mu}_j^{(t)}) \right\} \right) \\
&= -a_i \sum_{n=1}^{N_i} \mathbf{x}_{in} + N_i a_i \boldsymbol{\mu}_i + a_i \sum_{n=1}^{N_i} \mathbf{W}_i \mathbb{E}[\mathbf{z}_{in}] + \sum_{j\in\mathcal{B}_i} (\boldsymbol{\gamma}_{ij1} - \boldsymbol{\gamma}_{ji2}) \\
&\quad + 2\eta \sum_{j\in B_i} \left(\boldsymbol{\mu}_i - \frac{1}{2}(\boldsymbol{\mu}_i^{(t)} + \boldsymbol{\mu}_j^{(t)}) \right) \quad (\because \boldsymbol{\gamma}_{ij1}^{(t)} = \boldsymbol{\gamma}_{ij2}^{(t)}) \\
&= (N_i a_i + 2\eta|\mathcal{B}_i|)\, \boldsymbol{\mu}_i \\
&\quad - a_i \sum_{n=1}^{N_i} \mathbf{x}_{in} + a_i \sum_{n=1}^{N_i} \mathbf{W}_i \mathbb{E}[\mathbf{z}_{in}] + \sum_{j\in\mathcal{B}_i} (\boldsymbol{\gamma}_{ij1} - \boldsymbol{\gamma}_{ji2}) - \eta \sum_{j\in B_i} \left(\boldsymbol{\mu}_i^{(t)} + \boldsymbol{\mu}_j^{(t)} \right) \\
\therefore \boldsymbol{\mu}_i^{(t+1)} &= (N_i a_i + 2\eta|\mathcal{B}_i|)^{-1} \\
&\quad \cdot \left\{ a_i \sum_{n=1}^{N_i} (\mathbf{x}_{in} - \mathbf{W}_i \mathbb{E}[\mathbf{z}_{in}]) - 2\boldsymbol{\gamma}_i^{(t)} + \eta \sum_{j\in B_i} \left(\boldsymbol{\mu}_i^{(t)} + \boldsymbol{\mu}_j^{(t)} \right) \right\} \qquad \text{(B.35)}
\end{aligned}
$$

if we define $\boldsymbol{\gamma}_i = \sum_{\mathcal{B}_i} \boldsymbol{\gamma}_{ij}$ and apply Proposition 1.

Computing (B.4): We can apply the same approach to find the update for $\mathbf{W}_i$. We will use following properties of trace of matrix:

$$
\frac{\partial tr[\mathbf{A}\mathbf{B}\mathbf{A}^\top]}{\partial \mathbf{A}} = \mathbf{A}(\mathbf{B} + \mathbf{B}^\top), \qquad \text{(B.36)}
$$

$$
\frac{\partial tr[\mathbf{A}^\top \mathbf{B}]}{\partial \mathbf{A}} = \mathbf{B}. \qquad \text{(B.37)}
$$

Now,

$$\frac{\partial \mathcal{L}(\mathbf{\Phi}_i)}{\partial \mathbf{W}_i} = \frac{\partial}{\partial \mathbf{W}_i}\left[\sum_{n=1}^{N_i}\left\{\frac{a_i}{2}tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]\mathbf{W}_i^\top\mathbf{W}_i\right] - a_i\mathbb{E}[\mathbf{z}_{in}]^\top\mathbf{W}_i^\top(\mathbf{x}_{in}-\boldsymbol{\mu}_i)\right\}\right.$$
$$+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}\left(\boldsymbol{\lambda}_{ij1}^\top(\mathbf{W}_i-\boldsymbol{\rho}_{ij})+\boldsymbol{\lambda}_{ij2}^\top(\boldsymbol{\rho}_{ij}-\mathbf{W}_j)\right)$$
$$\left.+\frac{\eta}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{B}_i}(||\mathbf{W}_i-\boldsymbol{\rho}_{ij}||^2+||\boldsymbol{\rho}_{ij}-\mathbf{W}_j||^2)\right].$$

By applying the properties (B.36) and (B.37), we get

$$0 = \frac{\partial}{\partial \mathbf{W}_i}\left[\sum_{n=1}^{N_i}\left\{\frac{a_i}{2}tr\left[\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]\mathbf{W}_i^\top\mathbf{W}_i\right] - a_i\mathbb{E}[\mathbf{z}_{in}]^\top\mathbf{W}_i^\top(\mathbf{x}_{in}-\boldsymbol{\mu}_i)\right\}\right]$$
$$+\sum_{j\in\mathcal{B}_i}(\boldsymbol{\lambda}_{ij1}-\boldsymbol{\lambda}_{ji2})+2\eta\sum_{j\in B_i}\left(\mathbf{W}_i-\frac{1}{2}(\mathbf{W}_i^{(t)}+\mathbf{W}_j^{(t)})\right)$$
$$=\sum_{n=1}^{N_i}\left\{a_i\mathbf{W}_i\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]-a_i(\mathbf{x}_{in}-\boldsymbol{\mu}_i)\mathbb{E}[\mathbf{z}_{in}]^\top\right\}$$
$$+\sum_{j\in\mathcal{B}_i}(\boldsymbol{\lambda}_{ij1}-\boldsymbol{\lambda}_{ji2})+2\eta\sum_{j\in B_i}\left(\mathbf{W}_i-\frac{1}{2}(\mathbf{W}_i^{(t)}+\mathbf{W}_j^{(t)})\right) \tag{B.38}$$
$$\therefore \mathbf{W}_i^{(t+1)} = \left\{a_i\sum_{n=1}^{N_i}(\mathbf{x}_{in}-\boldsymbol{\mu}_i)\mathbb{E}[\mathbf{z}_{in}]^\top - 2\boldsymbol{\lambda}_i^{(t)}+\eta\sum_{j\in B_i}\left((\mathbf{W}_i^{(t)}+\mathbf{W}_j^{(t)})\right)\right\}$$
$$\cdot\left(a_i\sum_{n=1}^{N_i}\mathbb{E}[\mathbf{z}_{in}\mathbf{z}_{in}^\top]+2\eta|\mathcal{B}_i|\mathbf{I}\right)^{-1} \tag{B.39}$$

as we define $\boldsymbol{\lambda}_i = \sum_{\mathcal{B}_i}\boldsymbol{\lambda}_{ij}$ and apply Proposition 1.

Appendix C
Bregman ADMM (B-ADMM)

In this appendix, we briefly review a generalization of the ADMM algorithm using Bregman divergence [140] proposed by Wang and Banerjee [6]. The Bregman divergence is defined as following.

Definition 1. *(Bregman [140], Banerjee, et al. [98]) Let $\phi : \Omega \to \mathbb{R}$ be a real-valued, continuously differentiable, strictly convex function where Ω being a closed convex set. Then, the Bregman divergence between two points $\mathbf{p}, \mathbf{q} \in \Omega$ is defined as*

$$B_\phi(\mathbf{p}, \mathbf{q}) = \phi(\mathbf{p}) - \phi(\mathbf{q}) - \langle \phi(\mathbf{q}), \mathbf{p} - \mathbf{q} \rangle \tag{C.1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

When $\phi(\mathbf{p}) = \langle \mathbf{p}, \mathbf{p} \rangle$, the Bregman divergence becomes squared Euclidean distance [98]. When ϕ is the negative entropy, i.e., $\phi(\mathbf{p}) = \sum_{d=1}^{D_p} \mathbf{p}_d \log_2 \mathbf{p}_d$, where $\mathbf{p}_d$ denotes d-th element of $\mathbf{p} \in \mathbb{R}^{D_p}$ and $\sum_{d=1}^{D_p} \mathbf{p}_d = 1$, the Bregman divergence becomes Kullback-Leibler divergence [96].

As explained in Appendix A, we use squared norm for ensuring strict convexity in the standard ADMM optimization. In Bregman ADMM [6], we replace the squared norm with the Bregman divergence. We assume $B_\phi(\mathbf{c} - \mathbf{Ax}, \mathbf{By})$ is well defined. It is straightforward to see that the squared Euclidean norm is a special case of Bregman divergence when $\phi(\mathbf{p}) = \langle \mathbf{p}, \mathbf{p} \rangle$ as

$$B_\phi(\mathbf{c} - \mathbf{Ax}, \mathbf{By}) = \langle \mathbf{c} - \mathbf{Ax}, \mathbf{c} - \mathbf{Ax} \rangle - \langle \mathbf{By}, \mathbf{By} \rangle - \langle \mathbf{c} - \mathbf{Ax} - \mathbf{By}, 2\mathbf{By} \rangle \tag{C.2}$$

$$= \langle \mathbf{c} - \mathbf{Ax}, \mathbf{c} - \mathbf{Ax} \rangle + \langle \mathbf{By}, \mathbf{By} \rangle - 2 \cdot \langle \mathbf{c} - \mathbf{Ax}, \mathbf{By} \rangle \tag{C.3}$$

$$= \|\mathbf{c} - \mathbf{Ax} - \mathbf{By}\|^2 \tag{C.4}$$

$$= \|\mathbf{Ax} + \mathbf{By} - \mathbf{c}\|^2. \tag{C.5}$$

The augmented Lagrangian A.5 becomes

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}) + \rho B_\phi (\mathbf{c} - \mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{y}) . \tag{C.6}$$

However, we cannot optimize C.6 alternatingly as in the standard ADMM case since the Bregman divergence is not necessarily convex in the second argument. Specifically, one can minimize $\mathcal{L}_\rho(\mathbf{x}, \mathbf{y}^t, \boldsymbol{\lambda}^t)$ with the term $B_\phi \left(\mathbf{c} - \mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{y}^t\right)$ with respect to $\mathbf{x}$, but it can be challenging to minimize $\mathcal{L}_\rho(\mathbf{x}^{t+1}, \mathbf{y}, \boldsymbol{\lambda}^t)$ with respect to $\mathbf{y}$ because $B_\phi \left(\mathbf{c} - \mathbf{A}\mathbf{x}^{t+1}, \mathbf{B}\mathbf{y}\right)$ is not necessarily convex in $\mathbf{y}$. Wang and Banerjee [6] proposed to use the reverse of the parameters, i.e., $B_\phi \left(\mathbf{B}\mathbf{y}, \mathbf{c} - \mathbf{A}\mathbf{x}^{t+1}\right)$ to resolve this issue. The Bregman ADMM (B-ADMM) algorithm alternatingly minimizes using the following update steps:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{\lambda}^{t\top} \left(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}^t - \mathbf{c}\right) + \rho B_\phi \left(\mathbf{c} - \mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{y}^t\right) \tag{C.7}$$

$$\mathbf{y}^{t+1} = \arg\min_{\mathbf{y}} g(\mathbf{y}) + \boldsymbol{\lambda}^{t\top} \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y} - \mathbf{c}\right) + \rho B_\phi \left(\mathbf{B}\mathbf{y}, \mathbf{c} - \mathbf{A}\mathbf{x}^{t+1}\right) \tag{C.8}$$

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \rho \left(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{y}^{t+1} - \mathbf{c}\right) . \tag{C.9}$$

Futher generalization of the B-ADMM algorithm can be found in [6].

Appendix D

Message Passing ADMM

In this appendix, we briefly explain the message passing interpretation of the ADMM optimization algorithm [141, 142, 143]. The standard ADMM algorithm can be used to solve the following consensus-based convex optimization problem

$$\begin{aligned} \arg\min_{\mathbf{x},\mathbf{z}} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ s.t. \quad & \mathbf{x} = \mathbf{z} \end{aligned} \tag{D.1}$$

where $\mathbf{x}$, $\mathbf{z}$ are variables and f, g are proper convex functions. Then the augmented Lagrangian will be

$$\mathcal{L}_\rho = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^\top(\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}\|^2 \tag{D.2}$$

where y is the Lagrange multiplier and one can solve this with updates as

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \mathbf{y}^{t\top}\mathbf{x} + \frac{\rho}{2}\|\mathbf{x} + \mathbf{z}^t\|^2 \right) \tag{D.3}$$

$$\mathbf{z}^{t+1} = \arg\min_{\mathbf{z}} \left(g(\mathbf{z}) - \mathbf{y}^{t\top}\mathbf{z} + \frac{\rho}{2}\|\mathbf{x}^{t+1} - \mathbf{z}\|^2 \right) \tag{D.4}$$

$$\mathbf{y}^{t+1} = \mathbf{y}^t + \rho(\mathbf{x}^{t+1} - \mathbf{z}^{t+1}). \tag{D.5}$$

The message passing ADMM further modifies the objective (D.2) as

$$\mathcal{L}_\rho = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{u}^\top\rho(\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}\|^2 \tag{D.6}$$

by introducing a scaled dual variable $\mathbf{u} = \mathbf{y}/\rho$. Then the linear penalty term in the updates of ADMM can be incorporated into the squared regularizer term as

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{n}^t\|^2 \right) \tag{D.7}$$

$$\mathbf{z}^{t+1} = \arg\min_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{z} - \mathbf{m}^t\|^2 \right) \tag{D.8}$$

$$\mathbf{u}^{t+1} = \mathbf{u}^t + (\mathbf{x}^{t+1} - \mathbf{z}^{t+1}). \tag{D.9}$$

where we defined messages at iteration t as

$$\mathbf{m}^t = \mathbf{x}^t + \mathbf{u}^t \tag{D.10}$$

$$\mathbf{n}^t = \mathbf{z}^t - \mathbf{u}^t. \tag{D.11}$$

This is because the extra term $-(\rho/2) \cdot \|\mathbf{u}^t\|^2$ remained after completing the square can be ignored when minimizing with respect to $\mathbf{x}$ and $\mathbf{z}$. Note that by defining the messages, we can fully separate each coordinate update step for $\mathbf{x}^t$ and $\mathbf{z}^t$ to be independent from the other variable.

References

[1] S. Yoon and V. Pavlovic, "Distributed probabilistic learning for camera networks with missing data," in Bartlett *et al.* [144], pp. 2933–2941. v, 48, 50, 54, 56, 63, 64, 96

[2] B. Babagholami-Mohamadabadi, S. Yoon, and V. Pavlovic, "Mean field variational inference using Bregman ADMM for distributed camera network," in *Proceedings of the 9th International Conference on Distributed Smart Camera, Seville, Spain, September 8-11, 2015* (R. Carmona-Galán and Á. Rodríguez-Vázquez, eds.), pp. 209–210, ACM, 2015. v

[3] B. Gholami, S. Yoon, and V. Pavlovic, "Decentralized Approximate Bayesian Inference for Distributed Sensor Network," in Schuurmans and Wellman [145], pp. 1582–1588. v, 48

[4] C. Song, S. Yoon, and V. Pavlovic, "Fast ADMM algorithm for distributed optimization with adaptive penalty," in Schuurmans and Wellman [145], pp. 753–759. v, 64

[5] S. Yoon, M. Kapadia, P. Sahu, and V. Pavlovic, "Filling in the blanks: reconstructing microscopic crowd motion from multiple disparate noisy sensors," in *2016 IEEE Winter Applications of Computer Vision Workshops, WACV 2016 Workshops, Lake Placid, NY, USA, March 10, 2016*, pp. 1–9, IEEE Computer Society, 2016. v

[6] H. Wang and A. Banerjee, "Bregman alternating direction method of multipliers," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2816–2824, 2014. v, 43, 45, 105, 106

[7] J. Bennett and S. Lanning, "The Netflix Prize," in *In KDD Cup and Workshop in conjunction with KDD*, 2007. 2

[8] S. Zilberstein and S. Russell, "Anytime sensing, planning and action: A practical model for robot control," in *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1402–1407, 1993. 3

[9] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, and H. Zhu, "Decentralized learning for wireless communications and networking," in *Splitting Methods in Communication and Imaging, Science and Engineering* (R. Glowinski, S. Osher, and W. Yin, eds.), Springer, Mar. 2015. 3, 15

[10] R. Tron and R. Vidal, "Distributed Computer Vision Algorithms," *IEEE Signal Processing Magazine*, vol. 28, pp. 32–45, 2011. 3, 4, 16, 17, 19

[11] M. E. Tipping and C. M. Bishop, "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society, Series B*, vol. 61, pp. 611–622, 1999. 6, 25, 48, 61

[12] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, "A message-passing algorithm for multi-agent trajectory planning," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* (C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 521–529, 2013. 6, 68, 74, 75, 77, 78, 79, 80

[13] J. Bento, N. Derbinsky, C. Mathy, and J. S. Yedidia, "Proximal operators for multi-agent path planning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.* (B. Bonet and S. Koenig, eds.), pp. 3657–3663, AAAI Press, 2015. 6, 77, 79

[14] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989. 7, 8, 9, 10, 11, 12

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010. 9, 12, 13, 24, 45, 56, 57, 58, 68, 93

[16] R. Rockafellar, *Convex Analysis.* Princeton University Press, 1970. 10

[17] G. B. Dantzig and P. Wolfe, "Decomposition Principle for Linear Programs," *Operations Research*, vol. 8, pp. 101–111, 1960. 10

[18] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, September 1962. 10

[19] D. P. Palomar and M. Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, 2006. 10

[20] D. Bertsekas, *Nonlinear Programming.* Athena Scientific, 2nd ed., 1999. 10, 11

[21] J. Mota, *Communication-Efficient Algorithms For Distributed Optimization.* PhD thesis, Universidade Técnica de Lisboa Instituto Superior Técnico and Carnegie Mellon University, 2013. 11, 13, 14

[22] M. Balinski and P. Wolfe, eds., *Nondifferentiable Optimization*, vol. 3 of *Mathematical Programming Studies.* Springer Berlin Heidelberg. 11

[23] M. R. Hestenes, "Multiplier and Gradient Methods," *Journal of Optimization Theory and Applications*, vol. 4, no. 5, pp. 303–320, 1969. 11, 94

[24] M. J. D. Powell, "A method for nonlinear constraints in minimization problems," in *Optimization* (R. Fletcher, ed.), pp. 283–298, Academic Press, 1969. 11, 94

[25] R. Rockafellar, "Augmented Lagrangians and applications of the proximal point algorithm in convex programming," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 97–116, 1976. 12

[26] R. Glowinski and A. Marroco, "Sur l'approximation, par lments finis d'ordre un, et la rsolution, par pnalisation-dualit d'une classe de problmes de Dirichlet non linaires," *ESAIM: Mathematical Modelling and Numerical Analysis - Modlisation Mathmatique et Analyse Numrique*, vol. 9, no. R2, pp. 41–76, 1975. 12

[27] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17 – 40, 1976. 12

[28] T. Erseghe, D. Zennaro, E. DallAnese, and L. Vangelista, "Fast consensus by the alternating direction multipliers method," *IEEE Trans. Signal Processing*, vol. 59, no. 11, pp. 5523–5537, 2011. 13, 14

[29] M. DeGroot, "Reaching a consensus," *J. American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974. 13

[30] I. Schizas, A. Ribeiro, and G. Giannakis, "Consensus in ad hoc WSNs with noisy links - Part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008. 13

[31] I. D. Schizas, G. B. Giannakis, S. I. Roumeliotis, and A. Ribeiro, "Consensus in ad hoc WSNs with noisy links- Part II: Distributed estimation and smoothing of random signals," *IEEE Trans. Signal Processing*, vol. 56, no. 4, pp. 1650–1666, 2008. 13

[32] H. Zhu, G. Giannakis, and A. Cano, "Distributed in-network channel decoding," *IEEE Trans. Signal Processing*, vol. 57, no. 10, pp. 3970–3983, 2009. 13

[33] S.-J. Kim, E. DallAnese, and G. B. Giannakis, "Cooperative spectrum sensing for cognitive radios using Kriged Kalman filtering," *IEEE J. Selected Topics in Signal Processing*, vol. 5, no. 1, pp. 24–36, 2011. 13

[34] J. Mota, J. Xavier, P. Aguiar, and M. Puschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013. 13

[35] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-Based Distributed Support Vector Machines," *Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, 2010. 14, 24, 30

[36] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed Clustering Using Wireless Sensor Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, August 2011. 14, 15, 21, 23, 27, 54, 56, 61, 96, 100, 101

[37] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the 2005, American Control Conference, 2005.*, pp. 1859–1864 vol. 3, June 2005. 14

[38] M. J. Zaki and C.-T. Ho, eds., *Large-Scale Parallel Data Mining.* Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2000. 14

[39] H. Kargupta and P. Chan, eds., *Advances in Distributed and Parallel Knowledge Discovery.* The MIT Press, 2000. 14

[40] A. A. Freitas and S. H. Lavington, *Mining Very Large Databases with Parallel Processing*. Advances in Database Systems, Springer US, 2000. 14

[41] R. Bekkerman, M. Bilenko, and J. Langford, eds., *Scaling up machine learning : parallel and distributed approaches*. Cambridge University Press, 2012. 14, 15

[42] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Sixth Symposium on Operating System Design and Implementation (OSDI)*, 2004. 15

[43] D. Newman, A. Asuncion, P. Smyth, and M. Welling, "Distributed Algorithms for Topic Models," *Journal of Machine Learning Research*, pp. 1801–1828, 2009. 15

[44] S. Lee, J. K. Kim, X. Zheng, Q. Ho, G. A. Gibson, and E. P. Xing, "On Model Parallelization and Scheduling Strategies for Distributed Machine Learning," in *Advances in Neural Information Processing Systems 27*, 2014. 15

[45] H.-F. Yu, C.-J. Hsieh, H. Yun, S. Vishwanathan, and I. S. Dhillon, "A Scalable Asynchronous Distributed Algorithm for Topic Modeling," in *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (New York, NY, USA), pp. 1340–1350, ACM, 2015. 15

[46] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T.-Y. Liu, and W.-Y. Ma, "LightLDA: Big Topic Models on Modest Computer Clusters," in *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (New York, NY, USA), pp. 1351–1361, ACM, 2015. 15

[47] S. Zhu and Z. Ding, "Distributed cooperative localization of wireless sensor networks with convex hull constraint," *IEEE Transactions on Wireless Communications*, vol. 10, pp. 2150–2161, July 2011. 15

[48] J. A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Transactions on Signal Processing*, vol. 58, pp. 1847–1862, March 2010. 15

[49] A. H. Sayed, "Adaptive Networks," *Proceedings of the IEEE*, vol. 102, no. 4, 2014. 15

[50] B. Wang and K. J. R. Liu, "Advances in cognitive radio networks: A survey," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, pp. 5–23, Feb 2011. 15

[51] W. Jouini, C. Moy, and J. Palicot, "Decision making for cognitive radio equipment: analysis of the first 10 years of exploration," *EURASIP Journal on Wireless Communications and Networking*, 2012. 15

[52] R. Olfati-Saber, "Distributed Kalman Filtering for Sensor Networks," in *Decision and Control, 2007 46th IEEE Conference on*, pp. 5492 –5498, dec. 2007. 15, 16

[53] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in Bartlett *et al.* [144], pp. 1232–1240. 16

[54] A. Yang, S. Maji, C. Christoudias, T. Darrell, J. Malik, and S. Sastry, “Multiple-view Object Recognition in Band-limited Distributed Camera Networks,” in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, 30 2009-sept. 2 2009. 16

[55] R. J. Radke, “A Survey of Distributed Computer Vision Algorithms,” in *Handbook of Ambient Intelligence and Smart Environments* (H. Nakashima, H. Aghajan, and J. C. Augusto, eds.), Springer Science+Business Media, LLC, 2010. 16

[56] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, “Multi-camera people tracking with a probabilistic occupancy map,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 267–282, February 2008. 16, 19

[57] M. Taj and A. Cavallaro, “Distributed and decentralized multicamera tracking,” *IEEE Signal Processing Magazine*, vol. 28, pp. 46–58, May 2011. 16

[58] A. Wiesel and A. Hero, “Decomposable Principal Component Analysis,” *Signal Processing, IEEE Transactions on*, vol. 57, no. 11, pp. 4369–4377, 2009. 16

[59] S. V. Macua, P. Belanovic, and S. Zazo, “Consensus-based Distributed Principal Component Analysis in Wireless Sensor Networks,” in *Signal Processing Advances in Wireless Communications (SPAWC), 2010 IEEE Eleventh International Workshop on*, pp. 1–5, June 2010. 16

[60] R. Tron and R. Vidal, “Distributed Computer Vision Algorithms Through Distributed Averaging,” in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 57–63, June 2011. 16, 19, 33, 34, 36, 37, 49, 63, 71

[61] L. Xiao, S. Boyd, and S. Lall, “A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus,” in *International Conference on Information Processing in Sensor Networks*, pp. 63–70, April 2005. 16

[62] X. Wang and S. Wang, “Collaborative signal processing for target tracking in distributed wireless sensor networks,” *Journal of Parallel and Distributed Computing*, vol. 67, no. 5, pp. 501 – 515, 2007. 16

[63] B. Song, A. Kamal, C. Soto, C. Ding, J. Farrell, and A. Roy-Chowdhury, “Tracking and Activity Recognition Through Consensus in Distributed Camera Networks,” *Image Processing, IEEE Transactions on*, vol. 19, pp. 2564 –2579, oct. 2010. 16, 71

[64] P. Moreels and P. Perona, “Evaluation of Features Detectors and Descriptors based on 3D Objects,” *International Journal of Computer Vision*, vol. 73, pp. 263–284, July 2007. 18, 34, 50, 63

[65] M. Ozuysal, V. Lepetit, and P.Fua, “Pose Estimation for Category Specific Multiview Object Localization,” in *Conference on Computer Vision and Pattern Recognition*, (Miami, FL), June 2009. 18

[66] R. Tron and R. Vidal, “A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms,” in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, June 2007. 19, 35, 50, 63

[67] L. Sigal, A. O. Balan, and M. J. Black, "HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated HumanMotion," *International Journal of Computer Vision*, vol. 87, no. 1, pp. 4–27, 2009. 19

[68] S. Essid, X. Lin, M. Gowing, G. Kordelas, A. Aksay, P. Kelly, T. Fillon, Q. Zhang, A. Dielmann, V. Kitanovski, R. Tournemenne, N. E. O'Connor, P. Daras, and G. Richard, "A multimodal dance corpus for research into real-time interaction between humans in online virtual environments," in *ICMI Workshop On Multimodal Corpora For Machine Learning*, (Alicante, Spain), Nov. 2011. 19

[69] M. Alcântara, T. Moreira, and H. Pedrini, "Real-Time Action Recognition Based On Cumulative Motion Shapes," in *Acoustics, Speech and Signal Processing (ICASSP)*, 2014. 19

[70] J. Ferryman and A. Shahrokni, "Pets2009: Dataset and challenge," in *11th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2009. 19

[71] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple Object Tracking using K-Shortest Paths Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011. 19

[72] S. Zhang, E. Staudt, T. Faltemier, and A. Roy-Chowdhury, "A Camera Network Tracking (CamNeT) Dataset and Performance Baseline," in *IEEE Winter Conference on Applications of Computer Vision*, 2015. 20

[73] G. Denina, B. Bhanu, H. Nguyen, C. Ding, A. Kamal, C. Ravishankar, A. Roy-Chowdhury, A. Ivers, and B. Varda, "VideoWeb Dataset for Multi-camera Activities and Non-verbal Communication," in *Distributed Video Sensor Networks* (B. Bhanu, C. Ravishankar, A. Roy-Chowdhury, H. Aghajan, and D. Terzopoulos, eds.), Springer, 2010. 20

[74] A. Das, A. Chakraborty, and A. Roy-Chowdhury, "Consistent Re-identification In A Camera Network," in *European Conference on Computer Vision*, vol. 8690, pp. 330–345, 2014. 20

[75] D. Baltieri, R. Vezzani, and R. Cucchiara, "3D Body Model Construction and Matching for Real Time People Re-Identification," in *Proceedings of Eurographics Italian Chapter Conference 2010 (EG-IT 2010)*, (Genova, Italy), Nov. 2010. 20

[76] D. Baltieri, R. Vezzani, and R. Cucchiara, "SARC3D: a new 3D body model for People Tracking and Re-identification," in *Proceedings of the 16th International Conference on Image Analysis and Processing*, (Ravenna, Italy), pp. 197–206, Sept. 2011. 20

[77] D. Baltieri, R. Vezzani, and R. Cucchiara, "3DPes: 3D People Dataset for Surveillance and Forensics," in *Proceedings of the 1st International ACM Workshop on Multimedia access to 3D Human Objects*, (Scottsdale, Arizona, USA), pp. 59–64, Nov. 2011. 20

[78] M. Hofmann and D. Gavrila, "Multi-view 3D Human Upper Body Pose Estimation combining Single-frame Recovery, Temporal Integration and Model Adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 20

[79] M. Hofmann and D. Gavrila, "Multi-view 3D Human Pose Estimation in Complex Environment," *International Journal of Computer Vision*, vol. 96, no. 1, pp. 103–124, 2012. 20

[80] M. C. Liem and D. M. Gavrila, "A comparative study on multi-person tracking using overlapping cameras," in *International Conference on Computer Vision Systems (ICVS)*, vol. 7963 of *Lecture Notes in Computer Science*, pp. 203–212, 2013. 20

[81] S. Roweis and Z. Ghahramani, "A Unifying Review of Linear Gaussian Models," *Neural Computation*, vol. 11, pp. 305–345, 1999. 21

[82] A. R. Conn, N. I. M. Gould, and P. L. Toint, "A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds," *SIAM J. Numer. Anal.*, vol. 28, pp. 545–572, February 1991. 24

[83] R. M. Lewis and V. Torczon, "A Globally Convergent Augmented Lagrangian Pattern Search Algorithm for Optimization with General Constraints and Simple Bounds," *SIAM J. on Optimization*, vol. 12, pp. 1075–1089, April 2002. 24

[84] A. Ilin and T. Raiko, "Practical approaches to principal component analysis in the presence of missing values," *Journal of Machine Learning Research*, vol. 11, pp. 1957–2000, 2010. 28, 30, 33, 48

[85] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, pp. 137–154, 1992. 10.1007/BF00129684. 32

[86] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, April 1991. 35

[87] Y. Chikuse, *Statistics on Special Manifolds*, vol. 174 of *Lecture Notes in Statistics*. Springer, 1 ed., Feb. 2003. 37

[88] Z. Ghahramani and M. Beal, "Variational inference for Bayesian mixtures of factor analysers," in *Advances in Neural Information Processing Systems (NIPS)*, 2000. 40, 45

[89] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, Jan 2003. 40

[90] H. Attias, "A variational Bayesian framework for graphical models," in *Advances in Neural Information Processing Systems (NIPS)*, 2000. 40

[91] E. Fox, E. Sudderth, M. Jordan, and Willsky, "A sticky HDP-HMM with application to speaker diarization," *Annals of Applied Statistics*, vol. 5, pp. 1020–1056, 2011. 40

[92] J. Paisley and L. Carin, "Nonparametric factor analysis with beta process priors," in *International Conference on Machine Learning (ICML)*, 2009. 40

[93] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, 1999. 40

[94] M. Hoffman, D. Blei, C. Wang, and J. Paisly, "Stochastic Variational Inference," *Journal of Machine Learning Research*, vol. 14, pp. 1303–1347, 2013. 40

[95] S. Ahn, B. Shahbaba, and M. Welling, "Distributed Stochastic Gradient MCMC," *JMLR: W&CP*, vol. 32, 2014. Proceedings of The 31st International Conference on Machine Learning. 40

[96] S. Kullback, *Information Theory and Statistics.* John Wiley & Sons, 1959. 40, 105

[97] D. J. MacKay, *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003. 40

[98] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *Journal of Machine Learning Research*, vol. 6, pp. 1705–1749, Dec. 2005. 44, 105

[99] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *International Conference on Machine Learning (ICML)*, (Corvalis, Oregon, USA), pp. 209–216, June 2007. 44

[100] R. Liu, Z. Lin, F. De la Torre, and Z. Su, "Fixed-rank representation for unsupervised visual learning," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 598–605, 2012. 52

[101] L. Zhuang, H. Gao, Z. Lin, Y. Ma, X. Zhang, and N. Yu, "Non-negative low rank and sparse graph for semi-supervised learning," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012. 52

[102] E. Elhamifar, G. Sapiro, A. Y. Yang, and S. S. Sastry, "A Convex Optimization Framework for Active Learning," in *IEEE International Conference on Computer Vision, (ICCV)*, pp. 209–216, 2013. 52

[103] Z. Zeng, S. Xiao, K. Jia, T. Chan, S. Gao, D. Xu, and Y. Ma, "Learning by associating ambiguously labeled images," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 708–715, 2013. 52

[104] C. Wang, Y. Wang, Z. Lin, A. L. Yuille, and W. Gao, "Robust estimation of 3d human poses from a single image," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 2369–2376, 2014. 52

[105] K.-T. Lai, D. Liu, M.-S. Chen, and S.-F. Chang, "Recognizing complex events in videos by learning key static-dynamic evidences," in *Proceedings of the European Conference on Computer Vision (ECCV), 2014*, vol. 8691 of *Lecture Notes in Computer Science*, pp. 675–688, 2014. 52

[106] H. Boussaid and I. Kokkinos, “Fast and Exact: ADMM-Based Discriminative Shape Segmentation with Loopy Part Models,” in *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2014. 52

[107] O. Miksik, V. Vineet, P. Pérez, and P. H. S. Torr, “Distributed Non-Convex ADMM-inference in Large-scale Random Fields,” in *British Machine Vision Conference (BMVC)*, 2014. 52, 54

[108] B. He, H. Yang, and S. Wang, “Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities,” *Journal of Optimization Theory and Applications*, vol. 106, pp. 337–356, August 2000. 52, 55, 57, 58

[109] C. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006. 54

[110] B. He and X. Yuan, “On the $O(1/n)$ Convergence Rate of the Douglas-Rachford Alternating Direction Method,” *SIAM Journal of Numerical Analysis*, vol. 50, no. 2, pp. 700–709, 2012. 55

[111] T. Goldstein, B. O’Donoghue, S. Setzer, and R. Baraniuk, “Fast Alternating Direction Optimization Methods,” *SIAM Journal of Imaging Science*, vol. 7, no. 3, pp. 1588–1623, 2014. 55

[112] Y. Nesterov, “A method of solving a convex programming problem with convergence rate $o(1/k^2)$,” *Soviet Math. Dokl.*, vol. 27, pp. 372–376, 1983. 55

[113] H. Ouyang, N. He, L. Tran, and A. Gray, “Stochastic alternating direction method of multipliers.,” in *In Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013. 55

[114] T. Suzuki, “Dual averaging and proximal gradient descent for online alternating direction multiplier method,” in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013. 55

[115] Z. Xu, M. A. T. Figueiredo, and T. Goldstein, “Adaptive ADMM with spectral penalty parameter selection,” *CoRR*, vol. abs/1605.07246, 2016. 55

[116] R. T. Rockafellar, “Monotone operators and the proximal point algorithm,” *SIAM Journal on Control and Optimization*, vol. 14, p. 877, 1976. 56

[117] S. Ali, K. Nishino, D. Manocha, and M. Shah, *Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective.* Springer Publishing Company, Incorporated, 2013. 66, 69

[118] B. Zhan, D. Monekosso, P. Remagnino, S. Velastin, and L.-Q. Xu, “Crowd analysis: a survey,” *Machine Vision Applications*, vol. 19, pp. 345–357, 2008. 66, 69

[119] R. Mazzon and A. Cavallaro, “Multi-camera tracking using a Multi-Goal Social Force Model,” *Neurocomputing*, vol. 100, pp. 41–50, 2013. 66, 69, 70, 71

[120] S. Ali and M. Shah, “Floor Fields for Tracking in High Density Crowd Scenes,” in *ECCV*, 2008. 66, 69

[121] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll Never Walk Alone: Modeling Social Behavior for Multi-target Tracking," in *IEEE International Conference on Computer Vision*, 2009. 67, 69, 70

[122] G. Antonini, S. Martinez, M. Bierlaire, and J. Thiran, "Behavioral priors for detection and tracking of pedestrians in video sequences," *International Journal of Computer Vision*, vol. 96, pp. 159–180, 2006. 67

[123] A. J. D. Helbing, L. Buzna and T. Werner, "Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions," *Transportation Science*, vol. 39, no. 1, pp. 1–24, 2005. 67

[124] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People Tracking with Human Motion Predictions from Social Forces," in *IEEE International Conference on Robotics and Automation*, 2010. 67

[125] S. Avidan, Y. Moses, and Y. Moses, "Centralized and distributed multi-view correspondence," *Int. J. Comput. Vision*, vol. 71, pp. 49–69, Jan. 2007. 67

[126] J. Jacques Junior, S. Raupp Musse, and C. Jung, "Crowd analysis using computer vision techniques," *Signal Processing Magazine, IEEE*, vol. 27, pp. 66–77, Sept 2010. 69

[127] M. Kapadia, N. Pelechano, and J. Allbeck, *Virtual Crowds: Steps Toward Behavioral Realism.* MORGAN & CLAYPOOL, 2015. 69, 71

[128] M. Hu, S. Ali, and M. Shah, "Learning Motion Patterns in Crowded Scenes Using Motion Flow Field," in *ICPR*, 2008. 69

[129] M. Rodriguez, S. Ali, and T. Kanade, "Tracking in Unstructured Crowded Scenes," in *ICCV*, 2009. 69

[130] S. Pellegrini, A. Ess, M. Tanaskovic, and L. V. Gool, "Wrong turn - no dead end: a stochastic pedestrian motion model," in *International Workshop on Socially Intelligent Surveillance and Monitoring (SISM)*, 2010. 69

[131] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, May 1995. 69, 71, 80, 81

[132] S. Pellegrini and L. V. Gool, "Tracking with a mixed continuous-discrete conditional random field," *Computer Vision and Image Understanding*, January 2013. 70

[133] A. Bera and D. Manocha, "Realtime Multilevel Crowd Tracking using Reciprocal Velocity Obstacles," in *22nd International Conference on Pattern Recognition*, 2014. 70

[134] A. Bera, S. Kim, and D. Manocha, "Efficient Trajectory Extraction and Parameter Learning for Data-Driven Crowd Simulation," in *Graphics Interface*, 2015. 70

[135] J. van den Berg, M. C. Lin, and D. Manocha, "Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008. 70, 71

[136] A. Bera, N. Galoppo, D. Sharlet, A. Lake, and D. Manocha, "AdaPT: Real-time adaptive pedestrian tracking for crowded scenes," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 1801–1808, May 2014. 70

[137] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-aware large-scale crowd forecasting," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 2211–2218, June 2014. 70

[138] R. Eshel and Y. Moses, "Tracking in a dense crowd using multiple cameras.," *International Journal of Computer Vision*, vol. 88, no. 1, pp. 129–143, 2010. 70

[139] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example.," *Comput. Graph. Forum*, vol. 26, no. 3, pp. 655–664, 2007. 71

[140] L. M. Bregman, "The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, pp. 200–217, 1967. 105

[141] J. S. Yedidia, "Message-passing algorithms for inference and optimization," *Journal of Statistical Physics*, vol. 145, no. 4, pp. 860–890, 2011. 107

[142] J. S. Yedidia, Y. Wang, and S. C. Draper, "Divide and concur and difference-map BP decoders for LDPC codes," *IEEE Trans. Information Theory*, vol. 57, no. 2, pp. 786–802, 2011. 107

[143] N. Derbinsky, J. Bento, V. Elser, and J. S. Yedidia, "An improved three-weight message-passing algorithm," *CoRR*, vol. abs/1305.1961, 2013. 107

[144] P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, 2012. 109, 112

[145] D. Schuurmans and M. P. Wellman, eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, AAAI Press, 2016. 109