Distributed Probabilistic Learning for Camera Networks

by

Sejong Yoon and Vladimir Pavlovic

Rutgers, The State University of New Jersey
Piscataway, NJ 08854
\{sjyoon, vladimir\}@cs.rutgers.edu
Abstract

Probabilistic approaches to computer vision typically assume a centralized setting, with the algorithm granted access to all observed data points. However, many problems in wide-area surveillance can benefit from distributed modeling, either because of physical or computations constraints. In this work we present an approach to estimation and learning of generative probabilistic models in a distributed context. In particular, we show how traditional centralized models, such as probabilistic principal component analysis (PPCA), can be learned when the data is distributed across a network of sensors. We demonstrate the utility of this approach on the problem of distributed affine structure from motion (SfM). Our experiments suggest that the accuracy of the accuracy of the learned probabilistic structure and motion models rivals that of traditional centralized factorization methods.
1 Introduction

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, 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 views, one would like its local computational model to reflect a general 3D appearance of the object visible by other cameras in the network.

A number of distributed algorithms have been proposed to address the problems such as calibration, pose estimation, tracking, object and activity recognition in large camera networks [1–3]. In order to deal with high dimensionality of vision problems, distributed latent space search such as decentralized variants of PCA have been studied in [4, 5]. A more general framework using distributed least squares [6] based on distributed averaging of sensor fusions [7] 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 [8, 9]. 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.

In this work we propose a distributed consensus learning approach for parametric probabilistic models with latent variables. We assume that each node in a network can observe only a fraction of the data (e.g., object views in camera networks). 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 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 [10], and we extend to deal with generalized linear Gaussian models [11]. Unlike other recently proposed decomposable Gaussian graphical models [4, 12], 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 2, we first explain the general distributed probabilistic model. Section 3 shows how D-PPCA can be formulated as a special case of the probabilistic framework. We then explain how D-PPCA can be modified for the application in affine SfM. In Section 5, 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 6.

2 Distributed Probabilistic 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.

2.1 Centralized Setting

Let \( X = \{x_n | x_n \in \mathbb{R}^D\} \) be a set of iid multivariate data points with the corresponding latent variables \( Z = \{z_n | z_n \in \mathbb{R}^M\} \), \( n = \{1, 2, ..., N\} \). Our model is a joint density defined on \((x_n, z_n)\) with a global parameter \( (x_n, z_n) \sim p(x_n, z_n | \theta) \), as depicted in Fig. 1a. 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 \( p(z_n | x_n, \theta) \) is estimated, and (b) M-step: parametric optimization \( \hat{\theta} = \arg \max_\theta E_{Z|X} [\log p(X, Z | \theta)] \). It is important to point out that each posterior density estimate at point \( n \) depends solely on the corresponding measurement \( x_n \) and does not depend on any other \( x_k, k \neq n \). 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 \( X \) 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.

2.2 Distributed Setting

Let \( G = (V, E) \) be an undirected connected graph with vertices \( i, j \in V \) and edges \( e_{ij} = (i, j) \in E \) connecting the two vertices. Each \( i \)-th node is directly connected with 1-hop neighbors in \( B_i = \{j; e_{ij} \in E\} \). Suppose the set of data samples at \( i \)-th node is \( X_i = \{x_{in}; n = 1, ..., N_i\} \), where \( x_{in} \in \mathbb{R}^D \) is \( n \)-th measurement vector and \( N_i \) is the number of samples collected in \( i \)-th node. Likewise, we define the latent variable set for node \( i \) as \( Z_i = \{z_{in}; n = 1, ..., N_i\} \). As observed previously, each posterior estimation is decentralized. Learning the model parameter would be decentralized if each node had its own independent parameter \( \theta_i \). Still, the centralized model can be equivalently defined using the set of local parameters, with an additional constraint on their consensus, \( \theta_1 = \theta_2 = \cdots = \theta_{|V|} \). This is illustrated in Fig. 1b where the local node models are constrained using ties defined on the underlying graph. The simple consensus tying can be more conveniently defined using a set of auxiliary variables
Figure 1: Centralized, distributed and augmented models for probabilistic PCA.

\( \rho_{ij} \), one for each edge \( e_{ij} \) (Fig. 1c). This now leads to the final distributed consensus learning formulation, similar to [10]:

\[
\hat{\theta} = \arg \min_{\{\theta_i\}_{i \in V}} - \log p(X|\theta) \quad \text{s.t.} \quad \theta_i = \rho_{ij}, \rho_{ij} = \theta_j, i \in V, j \in B_i.
\]

(1)

This is a constrained optimization task that can be solved in a principal manner using the Alternating Direction Method of Multipliers (ADMM) [13–15]. ADMM iteratively, in a block-coordinate fashion, solves \( \max L(\cdot) \) on the augmented Lagrangian

\[
L(\theta, \rho, \lambda) = - \log p(X|\theta_1, \theta_2, ..., \theta_{|V|}, G) + \sum_{i \in V} \sum_{j \in B_i} \left\{ \lambda_{ij1}^T (\theta_i - \rho_{ij}) + \lambda_{ij2}^T (\rho_{ij} - \theta_j) \right\} + \frac{\eta}{2} \sum_{i \in V} \sum_{j \in B_i} \left\| \theta_i - \rho_{ij} \right\|^2 + \left\| \rho_{ij} - \theta_j \right\|^2
\]

(2)

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

3 Distributed Probabilistic PCA (D-PPCA)

We now apply the general distributed probabilistic learning explained above to the specific case of distributed PPCA. Traditional centralized formulation of probabilistic PCA (PPCA) [16] assumes that latent variable \( z_{in} \sim \mathcal{N}(z_{in}|0, I) \), with a generative relation

\[
x_{in} = W_i z_{in} + \mu_i + \epsilon_i,
\]

(3)
where \( \epsilon_i \sim \mathcal{N}(\epsilon_i|0, a_i^{-1}I) \) and \( a_i \) is the noise precision. Inference in this yields

\[
p(z_{in}|x_{in}) = \mathcal{N}(z_{in}|L_i^{-1}W_i^T(x_{in} - \mu_i), a_i^{-1}L_i^{-1}), \tag{4}
\]

where \( L_i = W_iW_i^T + a_i^{-1}I \). We can find optimal parameters \( W_i, \mu_i, a_i \) by finding the maximum likelihood estimates of the marginal data likelihood or by applying the EM algorithm on expected complete data log likelihood with respect to posterior density \( p(Z_i|X_i) \).

### 3.1 Distributed Formulation

The distributed algorithm developed in Section 2 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 \( W_i, \mu_i, a_i^{-1} \). The inference is accomplished locally in each node. The local parameter estimates are then computed using the consensus updates which 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 \( \Theta_i = \{W_i, \mu_i, a_i\} \) be the set of parameters for each node \( i \). The global constrained consensus optimization now becomes

\[
\min_{\{W_i, \mu_i, a_i; i \in V\}} -F(\Theta_i) \quad \text{s.t.} \quad \begin{align*}
W_i &= \rho_{ij}, & \rho_{ij} &= W_j, & i \in V, j \in B_i, \\
\mu_i &= \phi_{ij}, & \phi_{ij} &= \mu_j, & i \in V, j \in B_i, \quad \text{(5)}
\end{align*}
\]

where \( F(\Theta_i) = \sum_{n=1}^{N_i} \log p(x_{in}|W_i, \mu_i, a_i^{-1}) \). The augmented Lagrangian is

\[
\mathcal{L}(\Phi_i) = -F(\Theta_i) + \sum_{i \in V} \sum_{j \in B_i} (\lambda_{ij}^T(W_i - \rho_{ij}) + \lambda_{ij2}(\rho_{ij} - W_j)) + \sum_{i \in V} \sum_{j \in B_i} (\gamma_{ij1}(\mu_i - \phi_{ij}) + \gamma_{ij2}(\phi_{ij} - \mu_j))
\]

\[
+ \sum_{i \in V} \sum_{j \in B_i} (\beta_{i1}(a_i - \psi_{ij}) + \beta_{i2}(\psi_{ij} - a_j)) + \frac{\eta}{2} \sum_{i \in V} \sum_{j \in B_i} (||W_i - \rho_{ij}||^2 + ||\rho_{ij} - W_j||^2)
\]

\[
+ \frac{\eta}{2} \sum_{i \in V} \sum_{j \in B_i} (||\mu_i - \phi_{ij}||^2 + ||\phi_{ij} - \mu_j||^2) + \frac{\eta}{2} \sum_{i \in V} \sum_{j \in B_i} (a_i - \psi_{ij})^2 + (\psi_{ij} - a_j)^2 \quad \text{(6)}
\]

where \( \Phi_i = \{W_i, \mu_i, a_i, \rho_{ij}, \phi_{ij}, \psi_{ij}; i \in V, j \in B_i\} \) and \( \{\lambda_{ijk}\}, \{\gamma_{ijk}\}, \{\beta_{ijk}\} \) with \( k = 1, 2 \) are the Lagrange multipliers. The scalar value \( \eta \) gives us control over the convergence speed of the algorithm. With reasonably large positive \( \eta \), the overall optimization converges fairly quickly [10]. We will explore the converging behaviour with respect to various \( \eta \) in synthetic data experiments.

Just like in standard EM approach, we minimize the upper bound of \( \mathcal{L}(\Phi_i) \). Exploiting the posterior density in Eq. 4, we compute expected mean and variance of latent variables in each node as

\[
E[z_{in}] = L_i^{-1}W_i^T(x_{in} - \mu_i), \quad E[z_{in}z_{in}^T] = a_i^{-1}L_i^{-1} + E[z_{in}]E[z_{in}]^T. \quad \text{(7)}
\]
Algorithm 1 Distributed Probabilistic PCA (D-PPCA)

Require: For every node \( i \) initialize \( W^{(0)}_i, \mu_i^{(0)}, a_i^{(0)} \) randomly and set \( \lambda_i^{(0)} = 0, \gamma_i^{(0)} = 0, \beta_i^{(0)} = 0 \).

for \( t = 0, 1, 2, \ldots \) until convergence do

for all \( i \in V \) do

[E-step] Compute \( \mathbb{E}[z_{in}] \) and \( \mathbb{E}[z_{in}z_{in}^T] \) via (7).

[M-step] Compute \( W_i^{(t+1)}, \mu_i^{(t+1)}, a_i^{(t+1)} \) via (8,9,13).

end for

for all \( i \in V \) do

Broadcast \( W_i^{(t+1)}, \mu_i^{(t+1)}, \) and \( a_i^{(t+1)} \) to all neighbors of \( i \in B_i \).

end for

for all \( i \in V \) do

Compute \( \lambda_i^{(t+1)}, \gamma_i^{(t+1)}, \) and \( \beta_i^{(t+1)} \) via (10-12).

end for

end for

Maximization of the completed likelihood Lagrangian derived from Eq. 6 yields

$$W_i^{(t+1)} = \left\{ a_i \sum_{n=1}^{N_i} (x_{in} - \mu_i)\mathbb{E}[z_{in}]^T - 2\lambda_i^{(t)} + \eta \sum_{j \in B_i} \left( W_i^{(t)} + W_j^{(t)} \right) \right\}$$

$$\cdot \left( a_i \sum_{n=1}^{N_i} \mathbb{E}[z_{in}z_{in}^T] + 2\eta |B_i| I \right)^{-1},$$

$$\mu_i^{(t+1)} = \left\{ a_i \sum_{n=1}^{N_i} (x_{in} - \mu_i)\mathbb{E}[z_{in}] - 2\gamma_i^{(t)} + \eta \sum_{j \in B_i} \left( \mu_i^{(t)} + \mu_j^{(t)} \right) \right\} \cdot (N_i a_i + 2\eta |B_i|)^{-1},$$

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \eta \sum_{j \in B_i} \left\{ W_i^{(t+1)} - W_j^{(t+1)} \right\},$$

$$\gamma_i^{(t+1)} = \gamma_i^{(t)} + \eta \sum_{j \in B_i} \left\{ \mu_i^{(t+1)} - \mu_j^{(t+1)} \right\},$$

$$\beta_i^{(t+1)} = \beta_i^{(t)} + \eta \sum_{j \in B_i} \left\{ a_i^{(t+1)} - a_j^{(t+1)} \right\}.$$  

For \( a_i \), we solve the quadratic equation

$$0 = -\frac{N_i D}{2} + 2\eta |B_i| a_i^{(t+1)} + a_i^{(t+1)} \cdot \left\{ 2\beta_i^{(t)} - \eta \sum_{j \in B_i} \left( a_j^{(t)} + a_j^{(t)} \right) - \sum_{n=1}^{N_i} \mathbb{E}[z_{in}]^T W_i^{(t)} (x_{in} - \mu_i) \right\}$$

$$+ \frac{1}{2} \sum_{n=1}^{N_i} \left\{ ||x_{in} - \mu_i||^2 + tr \left[ \mathbb{E}[z_{in}z_{in}^T] W_i^{(t)} W_i^{(t)} \right] \right\}.$$  

The overall distributed EM algorithm for D-PPCA is summarized in Algorithm 1.
4 D-PPCA for Structure from Motion (SfM)

In this section, we consider a specific formulation of the modified distributed probabilistic PCA for application in affine SfM. In SfM, our goal is to estimate the 3D location of \( N \) points on a rigid object based on corresponding 2D 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 \cite{17}. Given a 2D (image coordinate) measurement matrix \( X \), of size \( 2 \cdot \# \text{frames} \times \# \text{points} \), the matrix is factorized into a \( 2 \cdot \# \text{frames} \times 3 \) motion matrix \( M \) and the \( 3 \times \# \text{points} \) 3D structure matrix \( S \). In the centralized setting this can be easily computed using SVD on \( X \). Equivalently, the estimates of \( M \) and \( S \) can be found using inference and learning in a centralized PPCA, where \( M \) is treated as the PPCA parameter and \( S \) is the latent structure. There we obtain additional estimates of the variance of structure \( S \), which are not immediately available from the factorization approach (although, they can be found).

However, the above defined \( (2 \cdot \# \text{frames} \times \# \text{points}) \) data structure of \( X \) is not amenable to distribution of different views (cameras, nodes), as considered in Section 3 of D-PPCA. Namely, D-PPCA assumes that the distribution is accomplished by splitting the data matrix \( X \) into sets of non-overlapping columns, one for each node. Here, however, we seek to distribute the rows of matrix \( X \), 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 \( X^T \). 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

\[
X_i^T = W \times Z_i + E_i
\]

where \( X_i^T \) is the matrix of image coordinates of all points in node (camera) \( i \) of size \( \# \text{points} \times 2 \cdot \# \text{frames} \) in node \( i \), \( W \) is the \( \# \text{points} \times 3 \) 3D structure (D-PPCA parameter) matrix and \( Z_i \) is the \( 3 \times 2 \cdot \# \text{frames} \) motion matrix of node \( i \).

One should note that we have implicitly assumed, in a standard D-PPCA manner, that each column of \( Z_i \) is iid and distributed as \( \mathcal{N}(0, I) \). However, each pair of subsequent \( Z_i \) columns represents one \( 3 \times 2 \) affine motion matrix. While those columns are not truly independent our experiments (as demonstrated in Section 5) show that this assumption is not detrimental in practice. Remaining task is simply following the same process we did to derive D-PPCA.

5 Experiments

In our experiments we first study the general convergence properties of the D-PPCA algorithm in a controlled synthetic setting. We then apply the D-PPCA to a set of SfM problems, both on synthetic and on real data.
5.1 Synthetic Data

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 50 dimensional 100 random samples from $N(0, 0.2 \cdot I)$. We assigned 20 samples equally to each node in a 5-nodes network connected with ring topology to find a 5 dimensional subspace. Our convergence criterion is the relative change in objective of Eq. 6 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.

Fig. 2a shows the convergence curve of D-PPCA for various $\eta$ values. As one can easily see, all $\eta$ values lead to convergence within $10^2$ iterations. Moreover, the converged value is equivalent to centralized solution meaning we can achieve the same global solution using the distributed algorithm. This behavior matches results reported in [18]. Fig. 2b shows convergence curve 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. Fig. 2c depicts the result 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.

5.2 Affine Structure from Motion

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.
5.2.1 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 [6]. The cube is centered at the origin of the space and rotates 30 degrees counterclockwise. We extracted 8 cube points projected on each camera view every 6 degrees, i.e. each camera observed 5 frames. Cameras are placed on a skewed plane, making elevation along $z$-axis as shown in Fig. 3a. For all synthetic and real SfM experiments, we picked $\eta = 10$ and initialized $W_i$ matrix with feature point coordinates of the first frame visible in the $i$-th camera with some small 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 noise free case, D-PPCA for SfM always yielded the same performance as SVD-based SfM with near zero degree.

We also tested D-PPCA for SfM with noisy data. First, we generated 20 independent samples of all 30 frames with 10 different noise levels. Then we ran D-PPCA 20 times on each of the independent sample with different initialization and averaged the final structure estimates. As Fig. 3b 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.
Table 1: Caltech 3D Objects on Turntable dataset statistics and quantitative results. Green dots indicate feature points tracked with correspondence across all 30 frames.

<table>
<thead>
<tr>
<th>Object</th>
<th>BallSander</th>
<th>BoxStuff</th>
<th>Rooster</th>
<th>Standing</th>
<th>StorageBin</th>
</tr>
</thead>
<tbody>
<tr>
<td># Points</td>
<td>62</td>
<td>67</td>
<td>189</td>
<td>310</td>
<td>102</td>
</tr>
<tr>
<td># Frames</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

Subspace angle between centralized SVD SfM and D-PPCA (degree)

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Variance</th>
<th></th>
<th>Mean</th>
<th>Variance</th>
<th></th>
<th></th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.4848</td>
<td>0.4159</td>
<td></td>
<td>1.4397</td>
<td>0.4567</td>
<td></td>
<td>1.4767</td>
<td>0.9448</td>
<td>1.6924</td>
</tr>
<tr>
<td></td>
<td>2.6221</td>
<td>1.6924</td>
<td></td>
<td>0.4463</td>
<td>1.2002</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2.2 Real Data

For real data experiment, we first applied D-PPCA for SfM on the Caltech 3D Objects on Turntable dataset [19]. The dataset provides various objects rotating on a turntable under different lighting conditions. The views of most objects were taken every 5 degrees 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 [6]. We first extracted first 30 degree images of each object. We then used KLT [20] 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 1 shows the 5 objects and statistics of feature points we extracted from the objects. We used \(\delta = 10\) 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 1 as the mean and variance of 20 independent runs. Experimental results indicate existence of differences between the reconstructions obtained by 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 noticeable differences, as illustrated in Fig. 4. Moreover, re-projecting back to the camera coordinate space resulted in close matching with the tracked feature points, as shown in Fig. 4.

We also tested the utility of D-PPCA for SfM on the Hopkins155 dataset [21]. We adopted virtually identical experimental setting as in [6]. We collected 135 single-object sequences containing image coordinates of points and we simulated multi-camera setting by partitioning the frames sequentially and almost equally for 5 nodes and the network

\(^1\)http://www.digilab.uni-hannover.de/docs/manual.html
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 $10^{-3}$. Average maximum subspace angle between D-PPCA for SfM and SVD-based SfM for all objects was 3.97 degree with 7.06 degree variance. 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 orthogonal ambiguity issue. Moreover, more than 53% of the all objects yielded subspace angle below 1 degree, 77% of them yielded below 5 degree and more than 94% were less than 15 degree.

6 Discussion and Future Work

In this work 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 or topology. 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 Eq. 14. 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 iid assumption of individual samples to that of subsequent columns (i.e., full 3x2 motion matrices). Our additional experiments, not reported here, indicate no discernable utility of this approach. A more principled approach would be to define priors for motion matrices compatible with SE(3), using e.g., [22]. While appealing, the priors would render the overall model non-linear and would require additional algorithmic considerations, perhaps in the spirit of [1].
Figure 4: Example of estimated 3D points and structure
References


