Data-driven Distributed State Estimation and Behavior Modeling in Sensor Networks

Rui Yu[1], Zhenyuan Yuan[2], Minghui Zhu[2], Zihan Zhou[1]

***Abstract*— Nowadays, the prevalence of sensor networks has enabled tracking of the states of dynamic objects for a wide spectrum of applications from autonomous driving to environmental monitoring and urban planning. However, tracking real-world objects often faces two key challenges: First, due to the limitation of individual sensors, state estimation needs to be solved in a collaborative and distributed manner. Second, the objects' movement behavior model is unknown, and needs to be learned using sensor observations. In this work, for the first time, we formally formulate the problem of simultaneous state estimation and behavior learning in a sensor network. We then propose a simple yet effective solution to this new problem by extending the Gaussian process-based Bayes filters (GP-BayesFilters) to an online, distributed setting. The effectiveness of the proposed method is evaluated on tracking objects with unknown movement behaviors using both synthetic data and data collected from a multi-robot platform.**

I. Introduction

Estimating the states of dynamic phenomena has long been a central problem in robotics with numerous real world applications. Taking autonomous driving as an example, for the autonomous cars to safely and efficiently navigate in a cluttered environment (*e.g.*, city traffic), it is critical to estimate the movements of other traffic agents (*e.g.*, pedestrians, bicyclists, and human-driven vehicles) and predict their future trajectories. Recent advances in sensing technologies have enabled the estimation and understanding of traffic movements using streaming data collected from various sensors such as on-board cameras, radar, and LiDAR, as well as surveillance cameras.

However, current state estimation methods still face two major challenges in such real scenarios. *First*, the capability of individual sensor is limited by its perception range (*e.g.*, line-of-sight and field-of-view) and also subject to noises and errors. Thus, it is often critical for the sensors to fuse information of others (*e.g.*, via vehicle-to-vehicle and vehicle-to-infrastructure communication networks) for accurate estimation. In other words, state estimation needs to be solved in a collaborative and distributed manner. *Second*, the movement behavior of traffic agents is highly non-linear, influenced by both their internal goals and interactions with other agents. Usually the internal goals and interactions are unknown, thus the movement behavior needs to be learned online using sensor observations.

Rather surprisingly, the two challenges mentioned above have never been studied together. Therefore, the **first contribution of this work** is that we formally formulate the problem of simultaneous state estimation and behavior modeling in a sensor network, and discuss its relation to existing work in the literature. Here, we emphasize an *online* learning setting, *i.e.*, we do not assume access to any training data with noise-free movement states, which are often difficult to obtain in real applications. Instead, the behavior model must be learned and updated using noisy sensor observations.

The **second contribution of this work** is a simple yet effective method to tackle this new problem. Our method is based on Bayes filtering with Gaussian process models (GP-BayesFilters) [1]. While traditional Bayes filters assume known (*e.g.*, parametric) motion and observation models, GP-BayesFilters use Gaussian process regression [2] to learn such models from data. We choose GP models over other data-driven models (*e.g.*, neural networks) based on two considerations. *First*, GP as a non-parametric model can learn from small-scale data, which is a competitive advantage in online learning. *Second*, GP models generate state-dependent uncertainty estimates that consider both noise and regression uncertainty due to limited training data, allowing them to be readily incorporated into various forms of Bayes filters (*e.g.*, extended Kalman filters).

In this work, we extend the GP-based Bayes filters in two aspects: (i) We apply it to distributed state estimation by using the consensus algorithms [3], where each sensor node can share information with neighbors and update the state by fusing information received from its neighbors; (ii) We show how the GP motion and observation models can be learned from noisy and partial observations in an online fashion, without requiring training data containing sequences of ground truth states.

The effectiveness of the proposed method is verified by experiments on both synthetic data and data collected from a multi-robot platform. We show that our data-driven GP models outperform pre-defined (*i.e.*, linear) models in prediction and tracking of traffic agents with unknown movement behavior. And the GP models learned by fusing information from multiple sensors via consensus are better than those learned by each sensor individually. In the experiment on robot platform, we show the advantage of data-driven models over parametric models in handling un-modeled system dynamics. Finally, we demonstrate that, compared to more complicated models (*i.e.*, deep neural networks), the GP

[1]R. Yu and Z. Zhou are with College of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16802, USA {rzy54, zuz22}@psu.edu

[2]Z. Yuan and M. Zhu are with School of Electrical Engineering and Computer Science, Pennsylvania State University, University Park, PA 16802, USA {zqy5086, muz16}@psu.edu

This work is supported in part by a seed grant from Penn State Institute for Computational and Data Sciences. Z. Yuan and M. Zhu were supported by NSF grants ECCS-1710859 and CNS-1830390.

model is a better choice for learning from small-scale noisy data in an online fashion.

II. Related Work

A. State Estimation with Gaussian Process Models

Gaussian process state space models (GPSSMs) are often used in state estimation problems for filtering [1], [4] or smoothing [5]. By describing the unknown transition and observation functions using GP models, Ko *et al*. [1] and Deisenroth *et al*. [4] derive different forms of GP-based Bayes filters including GP-EKF, GP-UKF, GP-PF, and GP-ADF. Since exact GP regression is challenging for large datasets due to $O(n^3)$ time complexity, [6] proposes sparse spectrum Gaussian process (SSGP), which maps the input to a low-dimensional feature space for fast inference. The original GP-BayesFilters also require ground truth states to train the GP models. To alleviate this issue, [7] first determines the latent states from observations by introducing a Gaussian Process prior over the latent space. Our work further extends the GPSSMs in two different ways, by (i) introducing consensus algorithms to the GP-BayesFilters framework for distributed state estimation, and (ii) learning the GP models online, where the states are estimated from observations using Kalman filters.

B. Distributed State Estimation

Consensus [3], [8] is a major methodology for distributed state estimation over sensor networks. The basic idea is that each network node runs a local filter, and information is spread through consensus of neighboring nodes. Several consensus algorithms have been developed, including consensus on estimation (CE) [9], consensus on measurement (CM) [10], and consensus on information (CI) [11]. [12] combines several algorithms into a general class of consensus filters named Hybrid CMCI. The stability of Hybrid CMCI is further proven in [13]. The consensus algorithm we adopt falls in this general class. But unlike existing consensus filters which assume known state transition and observation models, we study data-driven approaches to learn such models.

C. Multi-agent Behavior Modeling

Our work is also related to the line of research which attempts to learn the movement behavior of traffic agents. To describe human movement dynamics and interactions, several methods [14], [15], [16] use the social force model [17]. In these works, the model parameters are learned offline using manually annotated trajectories. [18] identifies the model parameters through a series of carefully designed observation experiments. [19] learns pedestrian motion patterns with Gaussian Processes models. Recently, pedestrian behaviors are modeled using deep networks such as LSTM [20] and GAN [21]. However, these methods do not consider distributed state estimation, and rely on ground truth states to learn the models.

III. Problem Formulation

In this work, we address the problem of collaborative tracking and behavior modeling of dynamic objects via a sensor network. Specifically, we are interested in a system composed of n sensors. The communication network connecting the sensors is represented by an undirected graph $G = (V, E)$ where vertices $V = \{1, \dots, n\}$ correspond to the sensors and an edge $(i, i') \in E$ indicates that sensors i and i' can communicate. We denote the neighbors of sensor i as $N_i = \{i' | (i, i') \in E, i' \in V \setminus \{i\}\}$.

Consider an environment that consists of m targets, whose states are represented by $\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_m\}$. We assume each target $\boldsymbol{y}_j$ evolves according to the following *target motion model*:

$$\boldsymbol{y}_j^{t+1} = f_j^t(\boldsymbol{y}_1^t, \dots, \boldsymbol{y}_m^t) + \eta_j^t, \tag{1}$$

where $\eta_j^t \sim \mathcal{N}(0, \Sigma_{\eta_j})$ is the process noise.

At any time t, sensor i can obtain state measurements of a subset of the targets. Let $\mathcal{O}_j^t \subseteq \{1, \dots, n\}$ be the set of sensors which can observe target j at time t, we assume a general *sensor observation model*:

$$\boldsymbol{z}_{ij}^t = h_{ij}^t(\boldsymbol{y}_j^t) + \nu_{ij}^t, \quad \forall i \in \mathcal{O}_j^t, \tag{2}$$

where $\nu_{ij}^t \sim \mathcal{N}(0, \Sigma_{\nu_{ij}})$ is the measurement noise.

Problem 1: Suppose the system function f_j and output function h_{ij} are time-invariant but remain unknown. The task of each sensor i is to construct estimators of the target states $\{\hat{\boldsymbol{y}}_j^t\}$ in a distributed fashion, *i.e.*, using information only from its neighbors and the measurements $\{\boldsymbol{z}_{ij}^t\}$.

In the following, we take the practical task of estimating and modeling human movements in traffic scenes as an example to illustrate our problem formulation.

Example 1: Consider the scenario of tracking multiple traffic agents (*e.g.*, pedestrians). Each agent state $\boldsymbol{y}_j$ consists of the position $\boldsymbol{x}_j$ and velocity $\boldsymbol{v}_j$. The dynamics of the system can be described as follows:

$$\begin{aligned} \boldsymbol{x}_j^{t+1} &= \boldsymbol{x}_j^t + \boldsymbol{v}_j^t \cdot \Delta t + \eta_{j,\boldsymbol{x}}^t \\ \boldsymbol{v}_j^{t+1} &= \boldsymbol{v}_j^t + g(\boldsymbol{y}_1^t, \dots, \boldsymbol{y}_m^t; \theta_j) \cdot \Delta t + \eta_{j,\boldsymbol{v}}^t \end{aligned} \tag{3}$$

where Δt is the sampling period and g is defined below.

We assume the motion of each agent follows a simplified version of social force model (SFM) [17], a popular formula for modeling human movement behavior in traffic scenes. According to SFM, each agent wants to keep its desired velocity, but is influenced by the other agents:

$$\begin{aligned} \frac{d\boldsymbol{v}_j^t}{dt} &= g(\boldsymbol{y}_1^t, \dots, \boldsymbol{y}_m^t; \theta_j) \\ &= \frac{1}{\tau_j}(\boldsymbol{v}_j^* - \boldsymbol{v}_j^t) + \sum_{k=1}^{m} \alpha_j e^{\frac{-\|\boldsymbol{x}_j^t - \boldsymbol{x}_k^t\|}{\beta_j}} \frac{\boldsymbol{x}_j^t - \boldsymbol{x}_k^t}{\|\boldsymbol{x}_j^t - \boldsymbol{x}_k^t\|}. \end{aligned} \tag{4}$$

Here, $\boldsymbol{v}_j^*$ and τ_j are the desired velocity and relaxation time, respectively, whereas α_j and β_j denote the strength and range of the interaction force. We have $\theta_j = \{\boldsymbol{v}_j^*, \tau_j, \alpha_j, \beta_j\}$.

Finally, we can assume a simple sensor observation model in which each sensor takes direct measurement of the target locations:

$$\boldsymbol{z}_{ij}^t = \boldsymbol{x}_j^t + \nu_{ij}^t, \quad \forall i \in \mathcal{O}_j^t. \tag{5}$$

IV. Online Data-driven Distributed EKF

We now describe our technical approach to the proposed problem. Let $\mathcal{T}_i^t \triangleq \{j | i \in \mathcal{O}_j^t\}$ be the set of targets observed by sensor $i \in V$ at time t. Using the augmented state $\boldsymbol{y}^t \triangleq [(\boldsymbol{y}_1^t)^T, \ldots, (\boldsymbol{y}_m^t)^T]^T$ and local output $\boldsymbol{z}_i^t = [(\boldsymbol{z}_{i1}^t)^T, \ldots, (\boldsymbol{z}_{i|\mathcal{T}_i^t|}^t)^T]^T$, system (1) and output (2) become

$$\boldsymbol{y}^{t+1} = f(\boldsymbol{y}^t) + \eta^t, \qquad \boldsymbol{z}_i^t = h_i(\boldsymbol{y}^t) + \nu_i^t$$

where

$$f(\boldsymbol{y}^t) \triangleq \begin{bmatrix} f_1(\boldsymbol{y}^t;\theta) \\ \vdots \\ f_m(\boldsymbol{y}^t;\theta) \end{bmatrix}, \qquad \eta^t \triangleq \begin{bmatrix} \eta_1^t \\ \vdots \\ \eta_m^t \end{bmatrix},$$

$$h_i(\boldsymbol{y}^t) \triangleq \begin{bmatrix} h_{i1}(\boldsymbol{y}_1^t) \\ \vdots \\ h_{i|\mathcal{T}_i^t|}(\boldsymbol{y}^t_{|\mathcal{T}_i^t|}) \end{bmatrix}, \quad \nu_i^t \triangleq \begin{bmatrix} \nu_{i1}^t \\ \vdots \\ \nu_{i|\mathcal{T}_i^t|}^t \end{bmatrix}.$$

Recall that the functions f and h_i are unknown in Problem 1. To overcome the limited knowledge on the system, we regard functions f and h_i as regression functions and estimate them by utilizing Gaussian process regression (GPR). In GPR, knowledge on the regression function is expressed as data set, which is a set of input-output examples of the regression function. At any time t, a set of input-output examples $D_{f,i}^t = \langle I_{f,i}^t, O_{f,i}^t \rangle$, $D_{h,i}^t = \langle I_{h,i}^t, O_{h,i}^t \rangle$ are stored by sensor i, where $I_{f,i}^t$ is a set of inputs and $O_{f,i}^t$ is a set of outputs of function f corresponding to the inputs. Likewise, $I_{h,i}^t$ is a set of inputs and $O_{h,i}^t$ is a set of outputs of function h_i corresponding to the inputs. We call $D_{f,i}^t$ and $D_{h,i}^t$ as data sets. They are initially empty, and are updated each time using estimates and outputs in an "online" fashion.

With the data-driven regression models, we can address Problem 1 in the Bayes filtering framework. We choose GP-EKF in this work and extend it to a distributed setting, but the same idea can be easily applied to other forms of GP-BayesFilters [1]. Algorithm 1 summarizes how the algorithm works in one episode. At any timestamp, each sensor node conducts state estimation given local information via GP-EKF (line 4), and then estimates are spread to neighboring nodes through a consensus algorithm (line 5). Finally, the knowledge on the regression functions (*i.e.*, datasets) is updated in lines 6-7.

In the following, we present the two key components of our method, namely the local GP-EKF and the consensus algorithm, in details.

A. Local GP-EKF

GPR is an algorithm to estimate an unknown function under the assumption that the regression function is a Gaussian process (GP). Formally, stochastic process f is Gaussian if $[f(\boldsymbol{y}_1), \cdots, f(\boldsymbol{y}_N)]^T$ is a multivariate Gaussian random variable, for any finite set of points $[\boldsymbol{y}_1, \cdots, \boldsymbol{y}_N]$ [22]. GPR could return a mean function of outputs. In our method, the functions f and h_i are estimated with GPR mean function.

Take the motion function f as an example. At time t, given the current dataset $D_{f,i}^{t-1} = \langle I_{f,i}^{t-1}, O_{f,i}^{t-1} \rangle$ and previous

Algorithm 1 Online Data-driven Distributed EKF

1: **Initialize**: target state $\hat{\boldsymbol{y}}_i^0$, information matrix Ω_i^0, data set $D_{f,i}^0 = \langle I_{f,i}^0, O_{f,i}^0 \rangle$, $D_{h,i}^0 = \langle I_{h,i}^0, O_{h,i}^0 \rangle$;
2: **for** $t = 1, 2, \ldots, T/\Delta t$ **do**
3: **Input**: $\hat{\boldsymbol{y}}_i^{t-1}$, Ω_i^{t-1}, $D_{f,i}^{t-1}$, $D_{h,i}^{t-1}$, measurement $\boldsymbol{z}_i^t$;
▷ ***Local GP-EKF*** (see Section IV-A)
4: Each sensor performs GP-EKF independently to obtain the prior information $(q_i^{t|t-1}, \Omega_i^{t|t-1})$ and the novel information $(\delta q_i^t, \delta \Omega_i^t)$;
▷ ***Hybrid Consensus*** (see Section IV-B)
5: Carry out consensus by iterating a number of regional averages on the two information pairs (prior and novel) and combine the fused pairs to obtain $(\hat{\boldsymbol{y}}_i^t, \Omega_i^t)$;
▷ ***Data set update***
6: $D_{f,i}^t = \langle [I_{f,i}^{t-1}, \hat{\boldsymbol{y}}_i^{t-1}], [O_{f,i}^{t-1}, \hat{\boldsymbol{y}}_i^t] \rangle$ for motion model;
7: $D_{h,i}^t = \langle [I_{h,i}^{t-1}, \hat{\boldsymbol{y}}^t], [O_{h,i}^{t-1}, \boldsymbol{z}_i^t] \rangle$ for observation model;
8: **Return:** $\hat{\boldsymbol{y}}_i^t$, Ω_i^t, $D_{f,i}^t$, $D_{h,i}^t$.
9: **end for**

state $\hat{\boldsymbol{y}}_i^{t-1}$, we want to predict the function output f_*. Under the assumption of GP, $O_{f,i}^{t-1}$ and f_* follow a joint Gaussian distribution

$$\mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(I_{f,i}^{t-1}, I_{f,i}^{t-1}) + \sigma_\epsilon^2 I & K(I_{f,i}^{t-1}, \hat{\boldsymbol{y}}_i^{t-1}) \\ K(\hat{\boldsymbol{y}}_i^{t-1}, I_{f,i}^{t-1}) & K(\hat{\boldsymbol{y}}_i^{t-1}, \hat{\boldsymbol{y}}_i^{t-1}) \end{bmatrix}\right)$$

where σ_ϵ represents the noise level of output data. Function K is kernel matrix (or covariance function) and the (a, b) element of $K(I_{f,i}^{t-1}, I_{f,i}^{t-1})$ is found by Gaussian kernel

$$\begin{aligned} K_{ab} &= k(I_{f,i}^{t-1}(a), I_{f,i}^{t-1}(b)) \\ &= \sigma_f^2 e^{-\frac{1}{2l_f^2}(I_{f,i}^{t-1}(a) - I_{f,i}^{t-1}(b))^T (I_{f,i}^{t-1}(a) - I_{f,i}^{t-1}(b))} \end{aligned}$$

where $I_{f,i}^{t-1}(a)$ and $I_{f,i}^{t-1}(b)$ denote a^{th}, b^{th} input data (column) of $I_{f,i}^{t-1}$ respectively. Parameter $\sigma_f \in \mathbb{R}$ represents the scale of the outputs, and $l_f \in \mathbb{R}$ is the length scale.

GPR mean function. By the standard rules for conditioning Gaussians, the posterior $p(f_* | \hat{\boldsymbol{y}}_i^{t-1}, D_{f,i}^{t-1})$ is also a Gaussian distribution with mean function μ_f

$$\mu_f(\hat{\boldsymbol{y}}_i^{t-1}, D_{f,i}^{t-1}) = O_{f,i}^{t-1}(K(I_{f,i}^{t-1}, I_{f,i}^{t-1}) + \sigma_\epsilon^2 I)^{-1} k_* \quad (6)$$

where $k_* = K(I_{f,i}^{t-1}, \hat{\boldsymbol{y}}_i^{t-1})$. And Jacobian matrix of μ_f can be described by

$$\begin{aligned} \frac{\partial \mu_f(\hat{\boldsymbol{y}}_i^{t-1}, D_{f,i}^{t-1})}{\partial \boldsymbol{y}} &= \frac{\partial O_{f,i}^{t-1}(K(I_{f,i}^{t-1}, I_{f,i}^{t-1}) + \sigma_\epsilon^2 I)^{-1} k_*}{\partial \boldsymbol{y}} \\ &= O_{f,i}^{t-1}(K(I_{f,i}^{t-1}, I_{f,i}^{t-1}) + \sigma_\epsilon^2 I)^{-1} \frac{\partial k_*}{\partial \boldsymbol{y}}. \end{aligned}$$

Similarly, one can also find GPR mean function $\mu_h(\hat{\boldsymbol{y}}_i^{t|t-1}, D_{h,i}^{t-1})$ and Jacobian matrix $\frac{\partial \mu_h}{\partial \boldsymbol{y}}(\hat{\boldsymbol{y}}_i^{t|t-1}, D_{h,i}^{t-1})$ for observation function h_i.

GPR covariance function. Covariance corresponding to the mean (6) can be calculated by GPR covariance function:

$$\begin{aligned} \Sigma_f(\hat{\boldsymbol{y}}_i^{t-1}, D_{f,i}^{t-1}) =& k(\hat{\boldsymbol{y}}_i^{t-1}, \hat{\boldsymbol{y}}_i^{t-1}) I \\ & - k_*(K(I_{f,i}^{t-1}, I_{f,i}^{t-1}) + \sigma_\epsilon^2 I)^{-1} (k_*)^T \end{aligned}$$

Algorithm 2 Local GP-EKF (Information filter form)

1: **Input**: $\hat{\boldsymbol{y}}_i^{t-1}$, Ω_i^{t-1}, $D_{f,i}^{t-1}$, $D_{h,i}^{t-1}$, $\boldsymbol{z}_i^t$;
▷ ***Local Prediction***
2: GPR mean function: $\hat{\boldsymbol{y}}_i^{t|t-1} = \mu_f(\hat{\boldsymbol{y}}_i^{t-1}, D_{f,i}^{t-1})$;
3: Jacobian matrix: $A_i^t = \frac{\partial \mu_f}{\partial \boldsymbol{y}}(\hat{\boldsymbol{y}}_i^{t-1}, D_{f,i}^{t-1})$;
4: GPR covariance: $Q_i^t = \Sigma_f(\hat{\boldsymbol{y}}_i^{t-1}, D_{f,i}^{t-1})$; $W_i^t = (Q_i^t)^{-1}$;
5: $\Omega_i^{t|t-1} = W_i^t - W_i^t A_i^t \left(\Omega_i^{t-1} + (A_i^t)^T W_i^t A_i^t\right)^{-1} (A_i^t)^T W_i^t$;
or: $\Omega_i^{t|t-1} = \left(A_i^t (\Omega_i^{t-1})^{-1} (A_i^t)^T + Q_i^t\right)^{-1}$;
6: Information vector: $q_i^{t|t-1} = \Omega_i^{t|t-1} \hat{\boldsymbol{y}}_i^{t|t-1}$;
▷ ***Local Correction***
7: Jacobian matrix: $C_i^t = \frac{\partial \mu_h}{\partial \boldsymbol{y}}(\hat{\boldsymbol{y}}_i^{t|t-1}, D_{h,i}^{t-1})$;
8: GPR covariance: $R_i^t = \Sigma_h(\hat{\boldsymbol{y}}_i^{t|t-1}, D_{h,i}^{t-1})$;
9: $\bar{\boldsymbol{z}}_i^t = \boldsymbol{z}_i^t - \mu_h(\hat{\boldsymbol{y}}_i^{t|t-1}, D_{h,i}^{t-1}) + C_i^t \hat{\boldsymbol{y}}_i^{t|t-1}$;
10: $\delta q_i^t = (C_i^t)^T (R_i^t)^{-1} \bar{\boldsymbol{z}}_i^t$;
11: $\delta \Omega_i^t = (C_i^t)^T (R_i^t)^{-1} C_i^t$;
12: **Output:** $q_i^{t|t-1}$, $\Omega_i^{t|t-1}$, δq_i^t, $\delta \Omega_i^t$.

Algorithm 3 Hybrid Consensus

1: **Input**: $q_i^{t|t-1}$, $\Omega_i^{t|t-1}$, δq_i^t, $\delta \Omega_i^t$;
▷ ***Consensus***
2: Prior info.: $q_i^{t|t-1}(0) = q_i^{t|t-1}$; $\Omega_i^{t|t-1}(0) = \Omega_i^{t|t-1}$;
3: Novel info.: $\delta q_i^t(0) = \delta q_i^t$; $\delta \Omega_i^t(0) = \delta \Omega_i^t$;
4: **for** $\ell = 0, \ldots, L-1$ **do**
5: Send $q_i^{t|t-1}(\ell)$, $\Omega_i^{t|t-1}(\ell)$, $\delta q_i^t(\ell)$, $\delta \Omega_i^t(\ell)$ to $i' \in N_i$;
6: Get $q_{i'}^{t|t-1}(\ell)$, $\Omega_{i'}^{t|t-1}(\ell)$, $\delta q_{i'}^t(\ell)$, $\delta \Omega_{i'}^t(\ell)$ from $i' \in N_i$;
7: $q_i^{t|t-1}(\ell+1) = \sum_{i' \in N_i \bigcup \{i\}} \pi^{ii'} q_{i'}^{t|t-1}(\ell)$;
8: $\Omega_i^{t|t-1}(\ell+1) = \sum_{i' \in N_i \bigcup \{i\}} \pi^{ii'} \Omega_{i'}^{t|t-1}(\ell)$;
9: $\delta q_i^t(\ell+1) = \sum_{i' \in N_i \bigcup \{i\}} \pi^{ii'} \delta q_{i'}^t(\ell)$;
10: $\delta \Omega_i^t(\ell+1) = \sum_{i' \in N_i \bigcup \{i\}} \pi^{ii'} \delta \Omega_{i'}^t(\ell)$;
11: **end for**
▷ ***Estimation***
12: $q_i^t = q_i^{t|t-1}(L) + \delta q_i^t(L)$;
13: $\Omega_i^t = \Omega_i^{t|t-1}(L) + \delta \Omega_i^t(L)$;
14: $\hat{\boldsymbol{y}}_i^t = (\Omega_i^t)^{-1} q_i^t$;
15: **Output:** $\hat{\boldsymbol{y}}_i^t$, Ω_i^t.

which represents covariance of the process Q_i^t. The same method is used for computing $\Sigma_h(\hat{\boldsymbol{y}}_i^{t|t-1}, D_{h,i}^{t-1})$ to represent R_i^t. Covariance matrices Q_i^t and R_i^t can be directly used in the Bayes filters.

As shown in Algorithm 2, at every timestamp, we independently perform GP-EKF for each sensor. To facilitate information fusion, we adopt the information filter form and estimate the information matrix $\Omega_i^t = (P_i^t)^{-1}$ rather than covariance matrix P_i^t. In particular, the pair $(q_i^{t|t-1}, \Omega_i^{t|t-1})$ called prior information is found by dynamic propagation (lines 2-6). Then, the current measurement $\boldsymbol{z}_i^t$ is used to find the pair $(\delta q_i^t, \delta \Omega_i^t)$ called novel information (lines 7-11).

B. Hybrid Consensus

As we discussed earlier, in practice, the observations of an individual sensor may be incomplete due to its perception range and occlusions, and also subject to noises and errors. Therefore, it is necessary to fuse information from different sensors for accurate state estimation.

In the literature, consensus is a widely used technique for distributed computations over networks. In this work, we adopt a special form of consensus with stability guarantee called hybrid consensus [13]. As shown in Algorithm 3, each node iteratively recalculates prior information $(q_i^{t|t-1}, \Omega_i^{t|t-1})$ and novel information $(\delta q_i^t, \delta \Omega_i^t)$ through communicating with its neighbors N_i, where $L \in \mathbb{N}$ is the number of iterations (lines 4-11). A set of weights $\pi_{ii'} \geq 0$ for $i' \in N_i$ is used for the recalculation, and is a convex combination, *i.e.*, $\sum_{i' \in N_i \bigcup \{i\}} \pi_{ii'} = 1$. Without loss of generality, we choose uniform weights in this work: $\pi_{ij} = 1/(|N_i| + 1)$ for $j \in N_i \bigcup \{i\}$. Then, the filtered estimate $\hat{\boldsymbol{y}}_i^t$ is computed by the consentaneous information (lines 12-14).

V. Experiments on Synthetic Data

In this section, we evaluate our method on synthetic data generated by simulating the movement and interaction of traffic agents (*e.g.*, pedestrians). Assume that the agents are tracked by a sensor network of four *localization sensors* and six *communication nodes*, which are randomly distributed within an 8.0×8.0 square area, as depicted in Fig. 1. Note that both types of nodes can process local information and exchange information with their linked neighbors, but only the localization sensors can make measurements about the targets. The effective range of a sensor is 4.0.

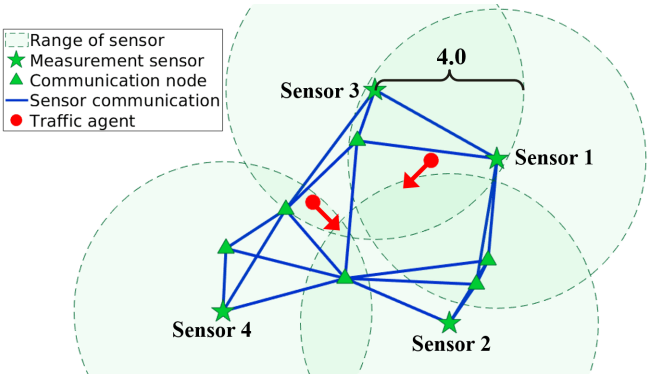


Fig. 1. Synthetic scenario of interaction between two traffic agents. The agents are tracked by a sensor network comprising four measurement sensors and six communication nodes.

A. Data Generation

Given m targets, we simulate the trajectory of each target using the social force model with parameters $\{\boldsymbol{v}_j^*, \tau, \alpha, \beta\}$, as described in Example 1. Each target has its own preferred velocity, but shares the remaining parameters with other targets. The default values are $\tau = 0.25, \alpha = 6, \beta = 5$.

We consider a typical case of $m = 2$. As shown in Fig. 1, the two targets start from different random positions in two 1.5×1.5 square areas and move to the center area simultaneously. The desired velocities are $\boldsymbol{v}_1^* = (1, -1)$ and $\boldsymbol{v}_2^* = (-1, -1)$, respectively. We add small noise $\eta \sim$

$\mathcal{N}(0, \sigma_\eta^2)$ with $\sigma_\eta = 0.001$ in Eq. (3).

To conduct simultaneous state estimation and behavior learning, we run the simulation to generate a large number of episodes of trajectories. The length of each episode is T with time step Δt, including $(T/\Delta t) + 1$ states $\boldsymbol{y}^t$ ($t \in \{0, 1, \ldots, T/\Delta t\}$). We set $T = 3$ and $\Delta t = 0.25$ in this experiment. Finally, the measurement model is given in Eq. (5) with noise $\nu \sim \mathcal{N}(0, \sigma_\nu^2)$ where $\sigma_\nu = 0.2$.

B. Motion Models

We compare the following motion models in EKF:

- **SFM**. The social force model as described in Eq. (3) and Eq. (4), *i.e.*, the true motion model. All parameters are assumed to be known, as described in Section V-A.
- **LIN**. A linear constant velocity model which predicts the state of each target independently

$$\boldsymbol{x}_j^{t+1} = \boldsymbol{x}_j^t + \boldsymbol{v}_j^t \cdot \Delta t, \qquad \boldsymbol{v}_j^{t+1} = \boldsymbol{v}_j^t.$$

  We assume the velocity of each target $\boldsymbol{v}_j^*$ is given. To accommodate the deviation from the true motion model, we use a higher process noise $\sigma_\eta = 0.1$ with LIN.
- **GP**. The Gaussian process regression model introduced in Section IV. The model hyperparameters are $\{\sigma_f, l_f, \sigma_\epsilon\}$, which are often estimated by maximum marginal likelihood [23]. To accommodate online learning, we manually tune the hyperparameters and set $\sigma_f = 1$, $l_f = 2$, and $\sigma_\epsilon = 0.5$ by default.

C. Experimental Protocol and Evaluation Metrics

In this experiment, we generate M episodes of trajectories for training, and N additional episodes for testing. In each episode, we initialize the state and the corresponding information matrix in Algorithm 1 with the first two observations as follows:

$$\hat{\boldsymbol{y}}_i^0 = \begin{pmatrix} \boldsymbol{z}_i^0 \\ \frac{\boldsymbol{z}_i^1 - \boldsymbol{z}_i^0}{\Delta t} \end{pmatrix}, \; \Omega_i^0 = \begin{bmatrix} \sigma_\nu^2 & & & \\ & \sigma_\nu^2 & & \\ & & \frac{2\sigma_\nu^2}{(\Delta t)^2} & \\ & & & \frac{2\sigma_\nu^2}{(\Delta t)^2} \end{bmatrix}^{-1}.$$

For GP, in the first training episode, we also compute the second state $\hat{\boldsymbol{y}}_i^1$ to initialize the data set: $D_{f,i}^1 = \langle \hat{\boldsymbol{y}}_i^0, \hat{\boldsymbol{y}}_i^1 \rangle$.

We use the root mean squared error (RMSE) of position to quantify the performance of each method. For each episode, the RMSE of the trajectory is computed as

$$RMSE = \frac{\Delta t}{mT} \sum_{t=1}^{T/\Delta t} \sum_{i=1}^{m} \|\boldsymbol{x}_i^t - \hat{\boldsymbol{x}}_i^t\|.$$

The averaged RMSE of all testing episodes with standard deviation is computed as the overall performance metrics.

D. Experiment Results

We have generated 250 episodes of trajectories and randomly split them into 200 training and 50 testing episodes. The default value of L is set to 4. We first compare the results obtained by running EKF on each sensor individually with those obtained by the consensus algorithm. Table I reports the average RMSE of different methods on the testing episodes. The results show that, due to the limited sensing range, running EKF on individual sensors yields large estimation errors. And GP performs worse than LIN, since GP relies on the observations to learn the motion model. With consensus, the performance improves significantly for all motion models. Further, GP achieves lower error than LIN, which suggests the importance of consensus in learning the motion model. We also note that, with consensus, all sensors have almost identical errors, indicating that the consensus algorithm converges with $L = 4$ for this sensor network topology.

TABLE I

TRACKING RMSE OF DISTRIBUTED EKF ON SYNTHETIC DATA.

		GP	LIN	SFM
Individual	Sensor 1	$\mathbf{1.24} \pm 0.23$	$\mathbf{1.29} \pm 0.72$	$\mathbf{0.37} \pm 0.23$
	Sensor 2	$\mathbf{0.81} \pm 0.40$	$\mathbf{0.66} \pm 0.51$	$\mathbf{0.26} \pm 0.14$
	Sensor 3	$\mathbf{0.68} \pm 0.18$	$\mathbf{0.47} \pm 0.22$	$\mathbf{0.16} \pm 0.06$
	Sensor 4	$\mathbf{1.80} \pm 0.43$	$\mathbf{1.02} \pm 0.51$	$\mathbf{0.26} \pm 0.06$
	Average	$\mathbf{1.13} \pm 0.31$	$\mathbf{0.86} \pm 0.49$	$\mathbf{0.26} \pm 0.12$
Consensus	Sensor 1	$\mathbf{0.14} \pm 0.04$	$\mathbf{0.16} \pm 0.03$	$\mathbf{0.14} \pm 0.10$
	Sensor 2	$\mathbf{0.14} \pm 0.04$	$\mathbf{0.16} \pm 0.03$	$\mathbf{0.14} \pm 0.10$
	Sensor 3	$\mathbf{0.14} \pm 0.04$	$\mathbf{0.16} \pm 0.03$	$\mathbf{0.14} \pm 0.10$
	Sensor 4	$\mathbf{0.14} \pm 0.04$	$\mathbf{0.16} \pm 0.03$	$\mathbf{0.14} \pm 0.10$
	Average	$\mathbf{0.14} \pm 0.04$	$\mathbf{0.16} \pm 0.03$	$\mathbf{0.14} \pm 0.10$

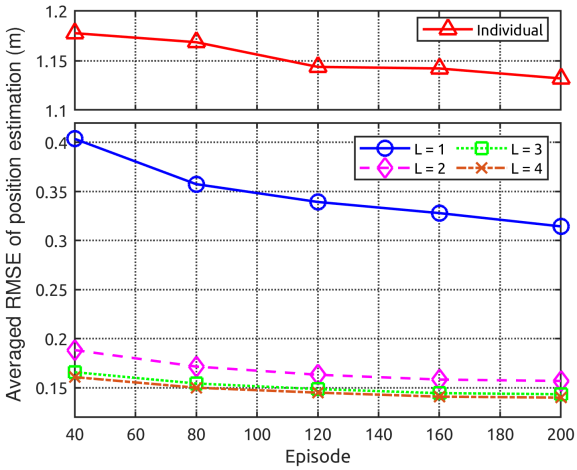


Fig. 2. Tracking performance of GP-EKF w.r.t. the number of training episodes and number of consensus iterations L.

Figure 2 plots the RMSE curves of GP-EKF w.r.t. two important parameters, namely the number of training episodes and L. As one can see, the error decreases as the number of training episodes increases, which indicates that GP is able to learn the movement behavior online. Further, comparing the RMSE curves for different choices of L, we can see that the consensus algorithm converges when $L \geq 3$.

We also conduct experiment to test the quality of the learned motion models, which are the critical components for Bayes filters. In this experiment, we provide each model with the ground truth state at $t = 0$, ask it to predict the states for $t = 1, \ldots, T/\Delta t$, and compute the average RMSE over all test sequences with standard deviation. The results are summarized in Table II. As can be seen, the GP (con.)

TABLE II

PREDICTION RMSE ON SYNTHETIC DATA.

	GP (ind.)	GP (con.)	LIN	SFM
S1	$\mathbf{1.38} \pm 0.36$	$\mathbf{0.24} \pm 0.10$	$\mathbf{1.30} \pm 0.11$	$\mathbf{0.004} \pm 0.002$
S2	$\mathbf{1.29} \pm 0.37$	$\mathbf{0.24} \pm 0.10$	$\mathbf{1.30} \pm 0.11$	$\mathbf{0.004} \pm 0.002$
S3	$\mathbf{0.73} \pm 0.19$	$\mathbf{0.24} \pm 0.10$	$\mathbf{1.30} \pm 0.11$	$\mathbf{0.004} \pm 0.002$
S4	$\mathbf{2.11} \pm 0.43$	$\mathbf{0.24} \pm 0.10$	$\mathbf{1.30} \pm 0.11$	$\mathbf{0.004} \pm 0.002$
Avg	$\mathbf{1.38} \pm 0.34$	$\mathbf{0.24} \pm 0.10$	$\mathbf{1.30} \pm 0.11$	$\mathbf{0.004} \pm 0.002$

TABLE III

TRACKING RMSE OF DISTRIBUTED EKF ON ROBOT DATA.

	GP	LIN	SFM
Sensor 1	$\mathbf{0.202} \pm 0.092$	$\mathbf{0.203} \pm 0.103$	$\mathbf{0.236} \pm 0.043$
Sensor 2	$\mathbf{0.204} \pm 0.094$	$\mathbf{0.206} \pm 0.106$	$\mathbf{0.236} \pm 0.043$
Sensor 3	$\mathbf{0.205} \pm 0.094$	$\mathbf{0.208} \pm 0.107$	$\mathbf{0.236} \pm 0.043$
Sensor 4	$\mathbf{0.202} \pm 0.089$	$\mathbf{0.202} \pm 0.098$	$\mathbf{0.236} \pm 0.042$
Average	$\mathbf{0.203} \pm 0.092$	$\mathbf{0.205} \pm 0.103$	$\mathbf{0.236} \pm 0.043$

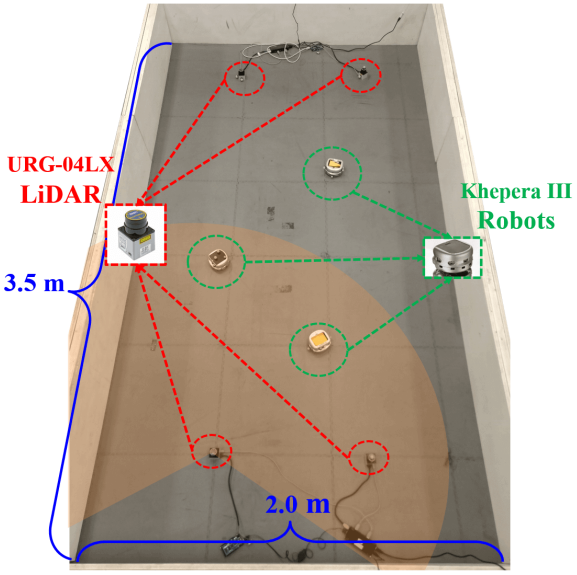


Fig. 3. The Khepera III robot platform.

models learned through running EKF with consensus are significantly better than the GP (ind.) models, which are learned by each sensor individually. And the GP (con.) models also significantly outperform the LIN models, verifying the effectiveness of our data-driven approach.

VI. EXPERIMENTS ON ROBOT PLATFORM

To assess the effectiveness of the proposed method on real system, we have also conducted experiments on a robot platform with three Khepera III robots [24].

A. Experiment Settings

As shown in Fig. 3, the arena is 3.5 meters long and 2.0 meters wide. The Khepera III robot has the size of 115mm×155mm×108mm and constructs differential drive dynamics. Each robot is programmed to move according to the social force model as described in Example 1. The model parameters are chosen as follows: $\tau = 1.0, \alpha = 0.24, \beta = 0.5$. The desired velocities of the robots are $\boldsymbol{v}_1^* = (0.04, 0.04)$m/s, $\boldsymbol{v}_2^* = (0.08, 0.08)$m/s, and $\boldsymbol{v}_3^* = (0.08, -0.08)$m/s. The scenario is designed to simulate the interactions among agents when crossing an intersection.

The locations of the robots are tracked by a VICON motion capture system with high accuracy. The system runs up to 1000 Hz, so we could also estimate the velocities based on the high-frequency location records. The locations and velocities obtained from VICON are considered as the ground truth states. As shown in Fig. 4, the robot locations obtained from VICON deviate from those predicted based on the ideal SFM model. The reason is that, in practice, the robot motion trajectory can be influenced by many unmodeled factors, such as frictional resistance, limitation of two-wheel motion, and so on.

Note that the ground truth states are used for evaluation purpose only. To conduct this experiment, we further obtain robot location measurement from four Hokuyo URG-04LX 2D LiDARs [25] placed in the arena. The Hokuyo LiDARs scan a 2D plane with an angle of 240 degrees and 0.36 angular resolution. The scan rate is 10 scans/sec and maximum range is 4.0 m. We design the arena to be enclosed by four walls to facilitate the LiDAR sensing. For each episode, we take 12 measurements with a time interval of 1.2s. The raw LiDAR measurements are first transferred into (x, y)-coordinates based on the position and orientation of each LiDAR. Then we rule out all points near the walls, apply DBSCAN clustering [26] to the remaining points, obtain the location measurements by computing the centers of the three largest clusters, and finally associate the measurements with the three robots according to their previous locations using the Hungarian algorithm. Based on the collected data, the measurement noise is set to $\sigma_\nu = 0.2$. In this experiment, we suppose each LiDAR can communicate with other two nearest LiDARs.

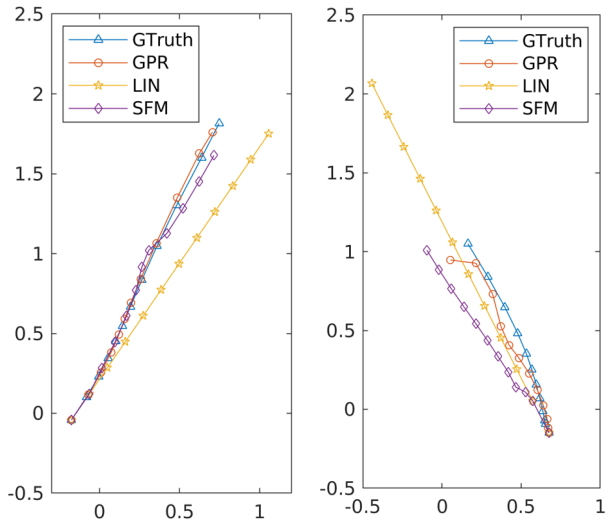
B. Experiment Results

For this experiment, we have collected 153 episodes and randomly split them into 128 training and 25 testing episodes. For all motion models, we adopt the same parameters as in Section V-B. These models are evaluated on the robot data using the same protocol as in Section V-C. Table III reports the tracking RMSE of different models in the distributed EKF framework, and Table IV reports the prediction results. As can be seen, the GP model not only outperforms the LIN model, but is also better than the "true" SFM model. As we mentioned above, real-world robot trajectories are influenced by other hard-to-model factors (*e.g.*, frictional resistance). The idealized SFM model ignores these factors while the GP model is able to learn them from observed real-world data. It explains why the performance of GP is on par with that of SFM on synthetic data but better than SFM on robotic data. It also demonstrates the reason why data-driven methods outperform idealized models on real-world experiments.

In Fig. 4, we further provide two representative examples of the trajectories predicted by different motion models. For better clarity, we only visualize the trajectories of one robot in each plot. In the designed scenario, every robot is trying

TABLE IV
PREDICTION RMSE ON ROBOT DATA.

	GP	LIN	SFM
Sensor 1	$\mathbf{0.125} \pm 0.024$	$\mathbf{0.688} \pm 0.045$	$\mathbf{0.242} \pm 0.044$
Sensor 2	$\mathbf{0.127} \pm 0.024$	$\mathbf{0.688} \pm 0.045$	$\mathbf{0.242} \pm 0.044$
Sensor 3	$\mathbf{0.127} \pm 0.024$	$\mathbf{0.688} \pm 0.045$	$\mathbf{0.242} \pm 0.044$
Sensor 4	$\mathbf{0.126} \pm 0.024$	$\mathbf{0.688} \pm 0.045$	$\mathbf{0.242} \pm 0.044$
Average	$\mathbf{0.126} \pm 0.024$	$\mathbf{0.688} \pm 0.045$	$\mathbf{0.242} \pm 0.044$



(a) Trajectory 1 (b) Trajectory 2

Fig. 4. Comparison of the predicted trajectories of different motion models.

to maintain the preferred velocity while avoiding collision with others. A common resulting phenomenon is that a robot would slow down or adjust the orientation if there is another robot in front of it. It is not surprising that LIN is neither able to predict the shift in direction (Fig. 4(a)) nor the change in speed (Fig. 4(b)). The SFM is able to model the "ideal" interactions among agents but ignores the aforementioned hard-to-model factors in real-world experiments. In contrast, GPR directly learns the multi-agent movement patterns from the noisy observations. As seen in Fig. 4, GPR produces the best predictions among all motion models. The experiment results thus show the advantage of online GP models in learning real-world motion patterns.

C. Comparison with State-of-the-Art Offline Prediction Model

Recently, deep neural networks, in particular long short-term memory (LSTM) models [20], are shown to achieve the state-of-the-art performance in predicting complex pedestrian movements. But unlike our problem setting in which the motion model (*i.e.*, GP) is learned online from noisy EKF estimations, existing work (*e.g.*, [20]) learns LSTM offline with human-annotated noise-free movement data.

In this experiment, we train the LSTM model on the robot data and compare it with our GP model in terms of their ability to *learn from small-scale noisy data*. Specifically, to prepare data to train LSTM offline, we use the same 128 training episodes and fuse the four LiDAR measurements

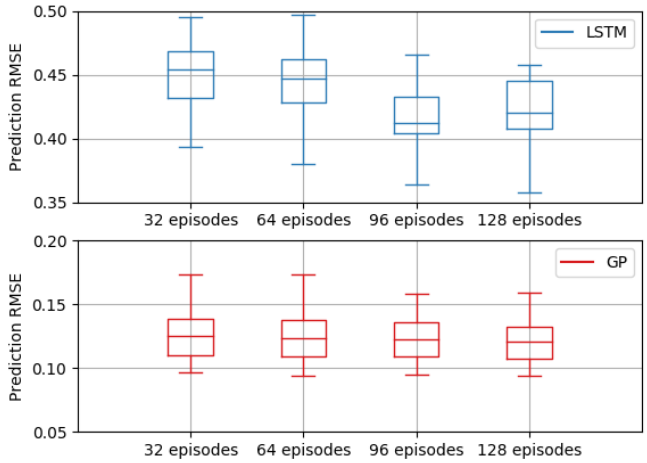


Fig. 5. Comparison of prediction RMSEs of LSTM and GP with different numbers of training episodes.

by using consensus on measurement (CM) method [10]. Following [20], we adopt an encoder-decoder sequence-to-sequence network architecture. The observed 2D points are first fed into a multi-layer perceptron (MLP) to obtain 64-dim feature embeddings. The encoder LSTM transforms the embedding into a 64-dim hidden state as the input of the decoder. The decoder LSTM transforms the 64-dim feature into a new 64-dim embedding. Then, another MLP converts decoder embedding into 2D vectors as the output prediction. The whole network is trained end-to-end using Adam [27] optimizer for 50 epochs with a learning rate of 1×10^{-3} and batch size of 8. The $L2$ loss is adopted to minimize the distance between the future measurements and predicted locations. The lengths of the input and output sequences are 2 and 10, respectively.

During testing, we follow the previous protocol to test the learnt LSTM model on same 25 testing episodes. In each episode, we provide LSTM with the ground truth state at $t = 0$. Based on the constant velocity assumption, we can estimate the location for $t = 1$ and utilize the trained sequence-to-sequence LSTM to predict the next 10 locations. Figure 5 shows the prediction results of the GP and LSTM models trained with varying number of episodes. As can be seen, GP significantly outperforms LSTM in this experiment. Besides, GP achieves more stable prediction performance even with a small number of training episodes (*e.g.*, 32 episodes), which demonstrates the advantage of GP in learning from small-scale noisy data.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have studied the problem of jointly estimating the states of dynamic objects and learning their movement behaviors in a sensor network. A simple yet effective solution is introduced, in which we extend the family of GP-BayesFilters to a distributed setting, and show that the GP models can be learned online with noisy sensor observations. Experiment results on both synthetic data and data simulated using a multi-robot platform to verify the effectiveness of our method.

While our problem formulation is quite general, assuming no knowledge about the motion and observation models, it still can be extended in multiple ways according to potential real-world scenarios. *First*, other noise models may be considered. For example, the observations may contain outliers, caused by sensor failures. *Second*, data association in multi-target tracking [28], *i.e.*, associating all observations which belong to the same target, must be addressed. And the problem becomes more challenging for a sensor network, as the association across views needs to be further computed [29]. *Third*, dynamic objects may change their movement behaviors over time, *e.g.*, a pedestrian stops to greet a friend on the sidewalk. We leave these directions to future research.

As to the technical approach, we plan to investigate fast methods for GP-based filters (*e.g.*, [6]) in order to maintain affordable computation with large-scale datasets. Finally, for the consensus filters, it is interesting to further study the cases where the communication topology and thus the weights are time-varying [8].

REFERENCES

[1] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, no. 1, pp. 75–90, 2009.

[2] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[3] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[4] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck, "Analytic moment-based Gaussian process filtering," in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML*, 2009, pp. 225–232.

[5] M. P. Deisenroth, R. D. Turner, M. F. Huber, U. D. Hanebeck, and C. E. Rasmussen, "Robust filtering and smoothing with Gaussian processes," *IEEE Trans. Automat. Contr.*, vol. 57, no. 7, pp. 1865–1871, 2012.

[6] Y. Pan, X. Yan, E. A. Theodorou, and B. Boots, "Prediction under uncertainty in sparse spectrum Gaussian processes with applications to filtering and control," in *Proceedings of the 34th International Conference on Machine Learning, ICML*, 2017, pp. 2760–2768.

[7] J. Ko and D. Fox, "Learning GP-bayesfilters via Gaussian process latent variable models," *Auton. Robots*, vol. 30, no. 1, pp. 3–23, 2011.

[8] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[9] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *46th IEEE Conference on Decision and Control, CDC*, 2007, pp. 5492–5498.

[10] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in *44th IEEE Conference on Decision and Control, CDC*. IEEE, 2005, pp. 8179–8184.

[11] G. Battistelli, L. Chisci, S. Morrocchi, and F. Papi, "An information-theoretic approach to distributed state estimation," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 12 477–12 482, 2011.

[12] G. Battistelli, L. Chisci, G. Mugnai, A. Farina, and A. Graziano, "Consensus-based linear and nonlinear filtering," *IEEE Trans. Automat. Contr.*, vol. 60, no. 5, pp. 1410–1415, 2015.

[13] G. Battistelli and L. Chisci, "Stability of consensus extended Kalman filter for distributed state estimation," *Automatica*, vol. 68, pp. 169–178, 2016.

[14] S. Pellegrini, A. Ess, K. Schindler, and L. J. V. Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *ICCV*, 2009, pp. 261–268.

[15] M. Luber, J. A. Stork, G. D. Tipaldi, and K. Arras, "People tracking with human motion predictions from social forces," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 464–469.

[16] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?" in *CVPR*, 2011, pp. 1345–1352.

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

[18] Y. Tamura, P. D. Le, K. Hitomi, N. P. Chandrasiri, T. Bando, A. Yamashita, and H. Asama, "Development of pedestrian behavior model taking account of intention," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 382–387.

[19] D. Ellis, E. Sommerlade, and I. D. Reid, "Modelling pedestrian trajectory patterns with Gaussian processes," in *12th IEEE International Conference on Computer Vision Workshops, ICCV Workshops*, 2009, pp. 1229–1234.

[20] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, F. Li, and S. Savarese, "Social LSTM: human trajectory prediction in crowded spaces," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 961–971.

[21] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: socially acceptable trajectories with generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018, pp. 2255–2264.

[22] D. J. C. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[23] Z. Chen and B. Wang, "How priors of initial hyperparameters affect Gaussian process regression models," *Neurocomputing*, vol. 275, pp. 1702–1710, 2018.

[24] F. Lambercy and J. Tharin, "Khepera III user manual," *K-Team*, Feburary 2013.

[25] Y. Mori, "Scanning laser range finder URG-04LX specifications," *Sentek Solution*, October 2005.

[26] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996, pp. 226–231.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[28] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *IEEE Control Systems Magazine*, vol. 29, no. 6, pp. 82–100, 2009.

[29] A. T. Kamal, J. H. Bappy, J. A. Farrell, and A. K. Roy-Chowdhury, "Distributed multi-target tracking and data association in vision networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1397–1410, 2016.