2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Distributed Mean-Field-Type Filter for Vehicle Tracking

Jian Gao and Hamidou Tembine

Abstract— **Particle filter is an effective tool for vehicle tracking. However, we need to maintain a large number of particles to keep a reasonable tracking accuracy for multi-target tracking in large scale state space. This paper proposes a new distributed mean-field-type filter to handle those noisy, partial-observed and high-dimensional data. The state space is decomposed and the particles are deployed locally and updated independently in the simplified subspaces. The filtering framework contains four operations: sampling, prediction, decomposition and correction. A mean-field term is included in the system dynamic so that the prediction is based on the previous state as well as its statistic distribution, which is estimated by a multi-frame learning procedure. The experiment on real data shows that our approach can achieve accurate tracking results with a small number of particles.**

I. Introduction

Vehicle tracking is a fundamental problem in Intelligent Traffic Systems (ITS). With the development of sensor technology, the observation data can be easily collected from many sources, such as inductive loops, video cameras and radars [1]. However, those data are often noisy and partially observed, which could make fatal conflicts and deteriorate the quality of ITS. Therefore, new learning, filtering and data assimilation techniques are required to process the noisy data collected from large scale traffic networks.

Vision-based vehicle tracking is a challenging problem because the quality of the video data is generally very poor [2]. The observation data is often contaminated by the noise from background interference, low resolution, lighting change, motion blur, and occlusion. Early work on filtering and signal estimation assumes Gaussian noise and linear systems. Kalman filter [3] uses a linear time-invariant dynamic model to estimate the optimal state, which can yield the exact conditional probability estimate with that assumption. The extensions EKF, UKF, EnKF and particle filters were designed to deal with nonlinear models and non Gaussian noise.

A dynamical system is said to be of mean-field type if it involves the probability measure of the state variable in the transition kernel to the next state [4]. The combination of observed data and dynamical models of mean-field type become a challenging problem since the system is huge and the observations are partial and noisy [5]–[8]. In [9] the authors observe that the approximation of most particle-based filters such as mean-field ensemble Kalman filter, approximate particle filter, particle swarm optimization based filter are not satisfactory for signal-observation dynamics of mean-field type. The main problems are the effective integration of high dimensional data and the control of error accumulation.

As an application, we implement our algorithm on video-based vehicle tracking, which is a popular task in ITS. Most previous work was focused on the detection side rather than the tracking side [2]. Despite well-designed appearance models [10], they use very simple motion model, or just randomly search around the previous vehicle positions [11]. However, fine-scale textures and well-designed features are not available in low quality videos. Our motivation is to design a robust filtering framework to estimate the vehicle positions accurately with noisy video data.

The contribution of this paper can be summarized as follows: (i) We propose a distributed mean-field-type filter (DMF) for vehicle tracking. The state space is decomposed into independent subspaces based on the foreground detection result. Particles are initialized as some good hypotheses and updated independently in each subspace. (ii) We provide a generic methodology to estimate the filtering distribution in four steps: sampling, prediction, decomposition and correction. This methodology can be applied to other high-dimensional networked systems with mean-field-type dynamic models and noisy observations. (iii) The mean-field term of system state is combined in the dynamic model to provide robust and accurate state prediction. A multi-frame learning procedure is designed to estimate the prior state distribution. The experimental results on real data have verified the usefulness and effectiveness of our approach.

The remainder of this paper is organized as follows: Section II introduces the system model and presents our distributed mean-field-type filter. Section III describes the design and implementation of our algorithm for vehicle tracking. Experimental results are also illustrated. Finally, we provide some concluding remarks in Section IV.

II. System Model

A dynamical system that involves the probability measure of the state variable in the transition kernel to the next state is called a mean-field-type system. We consider a process $(z_t)_t$ describing a high-dimensional dynamics of mean-field type with transition probability $\mathcal{K} := \mathbb{P}(z_{t+1} \in Z | z_t, \mathcal{L}_{z_t})$, while the process $(y_t)_t$ denotes the observation with conditional probability law $\mathbb{P}(y_t \in Y | z_t, \mathcal{L}_{z_t})$. Here $\mathcal{L}_{z_t}$ denotes the probability law of the state z_t. Our goal is to estimate the law of z_t based on the observation history $y_1, \ldots, y_t$. We introduce a nonlinear mean-field filter $m_t(Z) = \mathbb{P}(z_t \in$

This research is supported by U.S. Air Force Office of Scientific Research.
The authors are with Learning & Game Theory Laboratory, New York University Abu Dhabi, E-mail: tembine@nyu.edu

$Z|y_1, \ldots, y_t)$ to solve this problem. Once the conditional filtering distribution m_t is computed, we get a complete representation of the uncertainty. However, the implementation of m_t is not a trivial task, especially when the system is non-linear and the noise is non-Gaussian. It often requires some approximation by an empirical process $\hat{m}_t^N$. In high dimensional state space, we want to design a filtering approximation $\hat{m}_t^N$ such that the bound of error $\epsilon_N\sqrt{N}$ is independent of the cardinality of the entire network.

A. Sparse Decomposition of the State Space

The state space of the entire network is defined as a huge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} \neq \emptyset$. The set of vertices $\mathcal{V}$ is non-empty and arbitrary finite. The connections between the vertices are designed in a sparse manner $\|\mathcal{E}\| \ll \|\mathcal{V}\|^2$. For each signal-observation (z_t, y_t), there is a projection on the nodes $(z_{vt}, y_{vt})_{v\in\mathcal{V}}$. When the cardinality of $\mathcal{V}$ and $\mathcal{E}$ are very large, the system is considered as a high-dimension network, for example, the traffic network of the entire city. The set of nodes $\mathcal{V}$ can be decomposed into o components:

$$\mathcal{V} := \mathcal{V}_1 \oplus \mathcal{V}_2 \oplus \ldots \oplus \mathcal{V}_o \tag{1}$$

where $\mathcal{V}_j \cap \mathcal{V}_{j'} = \emptyset$ if $j \neq j'$. The signal space over the network is $\prod_{v\in\mathcal{V}} Z_v$, and the observation space is decomposed as $\prod_{v\in\mathcal{V}} Y_v$. Let $\mathcal{N}(v)$ be the neighborhood set of node v in the network, that is, $\mathcal{N}(v)$ contains all the vertices adjacent to v in $\mathcal{G}$. The transition at node v is sparse in the sense that only the neighborhood signal $(z_{wt})_{w\in\mathcal{N}(v)}$ and its distribution matter.

B. Distributed Mean-Field Filter Algorithm

At node v, the transition probability $\mathbb{P}(z_{v,t+1} \in Z_v|z_t, \mathcal{L}_{z_t})$ is the key term and determined by the neighborhood (augmented) state $(z_{wt})_{w\in\mathcal{N}(v)}$ and its distribution $\mathcal{L}_{z_{wt}}$. The observation at node v evolves according to the marginal conditional probability $\mathbb{P}(y_{vt} \in Y_v|z_{vt}, \mathcal{L}_{z_{vt}}) = l_{vt}(z_{vt}, \mathcal{L}_{z_{vt}}; y_{vt})$. The algorithm is as follows:

Initialization: T, N, $\hat{m}_0^N = m_0$. Here N is the number of particles and m_0 is the initial state distribution.

For time step t **from** 1 **to** T:

Sampling $\boldsymbol{S^N}$: Sample i.i.d $\hat{z}_t^i$ from $\hat{m}_t^N$. This is done by sampling independently from a product distribution $(\hat{m}_{t,|\mathcal{V}_1}^N, \ldots, \hat{m}_{t,|\mathcal{V}_o}^N)$ The operator S^N is therefore:

$$S^N m = \frac{1}{N}\sum_i \delta_{\hat{z}_t^i}; \tag{2}$$

Prediction $\boldsymbol{P}$: Predict the next state at node v for the i^{th} particle: $z_{v,t+1}^i \sim \mathcal{K}_{vt}(\hat{z}_t^i; \cdot)\mathcal{L}_{z_{vt}}$

Decomposition $\boldsymbol{D}$: Project a probability measure $m_{t,|\mathcal{V}}$ on the class of measures $m_{t,|\mathcal{V}_j}$, $(j = 1, \ldots, o)$, that are almost non-overlapping across zones, where

$$\mathcal{V} := \mathcal{V}_1 \oplus \mathcal{V}_2 \oplus \ldots \oplus \mathcal{V}_o$$

and $m_{t,|\mathcal{V}_j}$ is the marginal of m_t restricted to the zone $\mathcal{V}_j$, i.e., to the sub-space $\prod_{s\in\mathcal{V}_j} Z_s$, here s is the measurements or sensors in zone $\mathcal{V}_j$. For example, camera (s_1), radar (s_2) and inductive loop (s_3) form a detection system with three sources in zone $\mathcal{V}_j$. Thus, the decomposition operator yields:

$$Dm_t = (m_{t,|\mathcal{V}_1}, \ldots, m_{t,|\mathcal{V}_o}) \tag{3}$$

Correction $\boldsymbol{C_{t+1}}$: Compute the weight of the i^{th} particle w.r.t. the sensor s in zone $\mathcal{V}_j$:

$$w_{t+1|\mathcal{V}_j}^i = \frac{\prod_{s\in\mathcal{V}_j} l_{t+1}(z_{s,t+1}^i, \mathcal{L}_{z_{s,t+1}^i}; y_{s,t+1})}{\sum_{i'}\prod_{s\in\mathcal{V}_j} l_{t+1}(z_{s,t+1}^{i'}, \mathcal{L}_{z_{s,t+1}^{i'}}; y_{s,t+1})}$$

The estimated filtering distribution is set as the weighted sum of the particle states in each zone $\mathcal{V}_j$:

$$\hat{m}_{t+1,|\mathcal{V}_j}^N = \sum_{i=1}^N w_{t+1|\mathcal{V}_j}^i * \delta_{z_{\mathcal{V}_j,t+1}^i} \tag{4}$$

where $z_{\mathcal{V}_j,t+1}^i := (z_{s,t+1}^i)_{s\in\mathcal{V}_j}$.

The algorithm is distributed so that only observations (sensors) from the current zone $\mathcal{V}_j$ are used to update the filtering distribution $\hat{m}_{t+1,|\mathcal{V}_j}^N$.

As shown in Figure 1, the distributed mean-field filter (DMF) is therefore given by:

$$\hat{m}_{t+1}^N = C_{t+1}DPS^N\hat{m}_t^N =: \hat{O}_{t+1}^N\hat{m}_t^N. \tag{5}$$

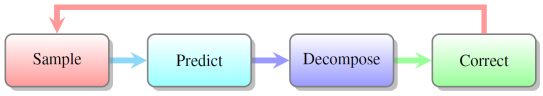


Fig. 1: Operators of the distributed mean-field filter algorithm

Theorem 1: The global error of the approximate mean-field filter is bounded by [12]:

$$\sup_{\|\phi\|\leq 1} E\|\phi(\hat{O}_{t+1}^N\hat{m}_t^N) - \phi(O_{t+1}m_t)\| \leq \epsilon_N \tag{6}$$

where the supremum is taken over test functions ϕ, and ϵ_N is independent of the time step t. Moreover, there is a constant $\kappa > 0$ such that $\sup_N \epsilon_N\sqrt{N} < \kappa o < +\infty$.

III. Implementation and Experimental Results

In this section we give the implementation of DMF in a real vehicle tracking scenario. The experiment results demonstrate the improvements in both accuracy and efficiency compared with the standard particle filters [13]. Figure 2 shows a map of the surveillance region. It is a subarea of the entire traffic network in Maryland, which covers about 30,000 square miles and contains 411 video cameras. For simplicity, we conducted the experiment on 12 cameras in this local area. The highway traffic sequences captured by these 12 video cameras are put together and generate a video with resolution 1920*1080. It contains 1376 frames at 15 frames per second. Our goal is to track multiple vehicles simultaneously in the surveillance regions and count the vehicle number on the road.

In this tracking problem, we implement DMF in a particle filter framework. Each particle is described by a vector with six coordinates $z_t = (px_t, py_t, sx_t, sy_t, vx_t, vy_t)^T$, which

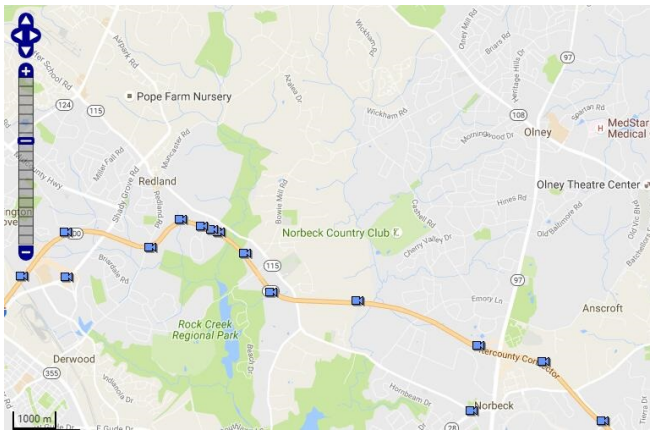


Fig. 2: Map of the surveillance region

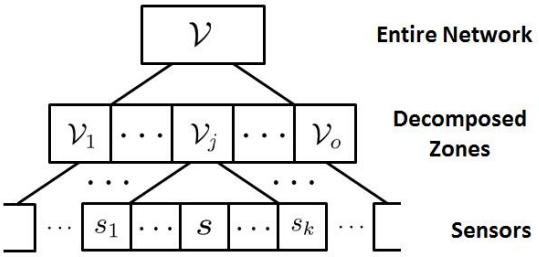


Fig. 3: High level decomposition. Decompose the entire network into o independent zones, each with k different sensors.

(a) (b)

Fig. 4: Tracking result. (a) Last frame (b) Prior distribution. Bright pixel indicates a high probability of vehicle existence.

are the vehicle's position, size, and velocity at frame t. The state variables $z_1, \ldots, z_t$ are estimated consecutively based on the observation sequence $y_1, \ldots, y_t$ generated by the sensors deployed in the surveillance region. For a traffic surveillance system in a city, thousands of gigabytes of data could be generated per second, which makes the analysis or even data transmission and storage quite challenging. What's more, the data is noisy and partially observed due to the low video quality and sparse distributed traffic cameras. We don't know what happens in the visual blind area between two adjacent cameras. What we can do is estimate and predict the state parameters based on the limited data nearby and make a reasonable guess concerning traffic conditions. Therefore, it is a good choice to decompose the network into relatively small and independent parts, and process the large amount of data in a distributed way.

There are two levels of decomposition. Figure 3 shows the high level decomposition. Suppose the entire space of the traffic network is $\mathcal{V}$, then we decompose it into o zones $\mathcal{V}_1, \ldots, \mathcal{V}_o$, e.g. o different roads, and in each zone there are k measurements $s_1, \ldots, s_k$, such as radars, cameras and loops. The observation of sensor s is $y_{s,t+1}$. Measurements of different sensors can be fused using particle filter [14]. On the low level, we decompose the state space based on the prior distribution. For example, in Figure 4b, the image domain can be decomposed into three lanes. We deploy the particles with high probability in the lanes and with low probability in the tree or sky areas, so as to reduce false alarms. Since vehicles in the same lane have interactions with each other and vehicles in different lanes are relatively independent, the particles are updated and resampled independently in each zone.

The state z could be position and velocity of one vehicle, the queuing length at intersections, or the traffic density within a local area, which depends on the specific applications. However, the initial state is unknown in most cases, which is quite different from many object tracking scenarios where the first-frame bounding box is given. There are two ways to handle this. The first is to make a random guess about the initial state, and put the particles uniformly in the state space. As the new observation sequence comes, the particles will evolve to be more and more accurate hypotheses. The other way is to start from some good initial hypotheses, which are obtained by a foreground detection mechanism. Our experiment has shown that starting from the vehicle detection hypotheses is much better than starting from a uniform distribution.

A. *Foreground Detection*

The domain of foreground detection is quite large [15] [16] [17] [18]. According to the previous research, a good foreground detection algorithm should have three properties. First, it should contain multiple complementary features, such as image intensity, color, gradient and texture. Second, the background model should be adaptive to the recent observed history, i.e. the foreground. Third, to identify the missing foreground-color pixels on targets, the spacial-time consistency should be exploited.

To generate the background model, we use the temporal median filter. We use the median value of each pixel in a constant time interval as the background. In our experiment, the background image is updated once every 25 frames by calculating the temporal median in the recent 250 frames. The background model is robust to mild illumination changes and the noise caused by the swaying trees. Then, we extract two complementary features from the video sequence: gray-scale intensity and gradient variation. Chromaticity variation [19] is not used because it is not distinctive in our case. For each point, the extracted feature values are compared with the background model. The different values indicate the probability that the pixel belongs to foreground. Finally,

to remove the false alarms caused by noise and identify the missing parts on a real target, we assign the pixel labels using loopy belief propagation (LBP) [20] under a Markov Random Field (MRF) optimization framework [21]. The idea is to exploit the spatial consistency and receive messages from neighboring pixels: for the current pixel $i \in I$, its probability of being foreground will increase if its four adjacent neighbors $j \in \mathcal{N}(i)$ are foreground pixels, and decrease if the neighbors are background. Let l be the label: 0 for background and 1 for foreground. The energy function of a label assignment is:

$$C(l) = \sum_{i \in I} \phi_i(l_i) + \sum_{i \in I} \sum_{j \in \mathcal{N}(i)} \psi_{ij}(l_i, l_j) \tag{7}$$

The data term $\phi_i(l_i)$ measures the cost of assigned label l_i to point i, which is given by the probability that i is assigned to a wrong label. The smoothness term $\psi_{ij}(l_i, l_j)$ measures the spacial inconsistency of the two labels l_i and l_j. If the neighboring pixels i and j have different labels, $\psi_{ij}(l_i, l_j) = 1$, otherwise it is 0. In each iteration, messages are propagated around the MRF. We choose the order of up, down, left, right for the message passing. By iteratively minimizing the cost function, we can get the optimized label assignment. The foreground detection result is shown in Figure 5.

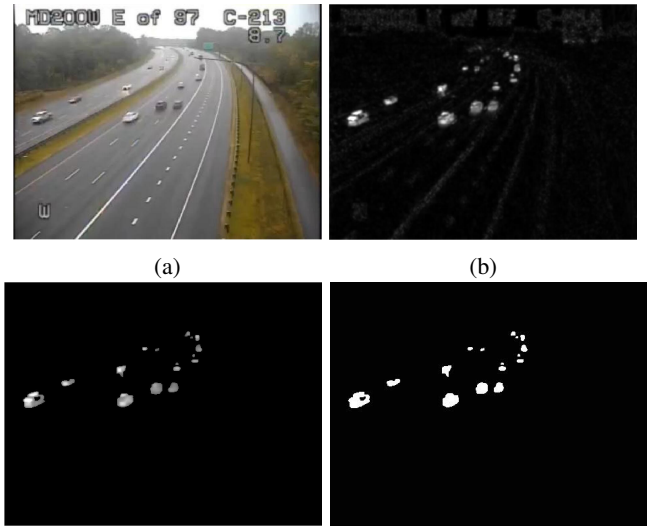


Fig. 5: Foreground Detection Result. (a) The original image. (b) Probability of being foreground before BP. (c) Probability of being foreground after BP. (d) Foreground labeling result.

B. Mean-Field Term

Since the foreground regions have a high probability of vehicle existence, it is reasonable to use the foreground detection result to initialize the tracking procedure. For the T detected foreground blobs, we decompose the image space into T independent zones and deploy N particles in each zone for the multi-target tracking task. Compared with uniformly scattering the particles in the whole state space, it is more effective to start from these good hypotheses. As new observations come in the next frame, particle weights are updated independently in each zone. A resampling step is triggered to discard the particles with too small weights, i.e. those deployed far away from the real targets, and thus put more particles near the target area. The particle state $z_t = (px_t, py_t, sx_t, sy_t, vx_t, vy_t)^T$ is evolved by the transition kernel $\mathcal{K}_t(z_t, \mathcal{L}_{z_t}; z_{t+1})$. The system dynamics is:

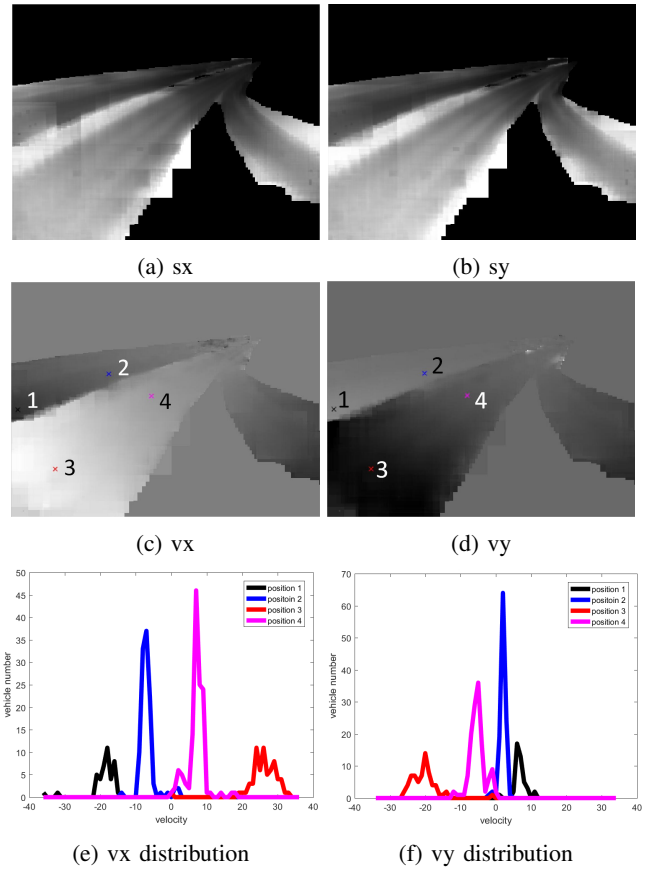
$$\begin{aligned}
px_{t+1} &= px_t + vx_t + n_{px} \\
py_{t+1} &= py_t + vy_t + n_{py} \\
sx_{t+1} &= sx_t * (E_{sx}(px_{t+1}, py_{t+1}) / E_{sx}(px_t, py_t) + n_{sx}) \\
sy_{t+1} &= sy_t * (E_{sy}(px_{t+1}, py_{t+1}) / E_{sy}(px_t, py_t) + n_{sy}) \\
vx_{t+1} &= \alpha vx_t + (1-\alpha) E_{vx}(px_{t+1}, py_{t+1}) + n_{vx} \\
vy_{t+1} &= \alpha vy_t + (1-\alpha) E_{vy}(px_{t+1}, py_{t+1}) + n_{vy}
\end{aligned} \tag{8}$$

where $n = (n_{px}, n_{py}, nsx, nsy, nvx, nvy)^T$ is the random Gaussian noise vector with variance $\sigma_i^2, i = 1, \ldots, 6$. E_{sx}, E_{sy}, E_{vx}, E_{vy} are the mean-field terms which represent the expected vehicle size (sx, sy) and velocity (vx, vy) at a specific position (px, py) in the image.

There are two reasons to use the mean-field terms. First, the existing approaches update the system state based on the previous state, which may amplify the error if the system model is inaccurate. For example, the vehicles may speed up when driving downhill and the drivers may slow down when it begins to rain, which makes our constant velocity assumption invalid. Therefore, we should design a dynamic system model to fit the current road and weather conditions with a certain distribution of the uncertainty. However, due to the complexity and variability of external factors, it is improper to just add some parameters and make a higher-order system. Instead, we designed a learning procedure and use the previous experience to guide our current task. This is what most drivers do: if the vehicle ahead slows down, he will also slow down. Second, there's a small part of the image that contains vehicles, i.e. the road zone, and most false alarms come from waving trees and moving clouds in the sky. Moreover, the vehicle's speed varies with the lanes (fast lanes and slow lanes), and the traffic flow is dense and slow near an intersection. So the distribution of vehicle position and speed should reflect these intuitions and be spatially distinct. In light of this, it is reasonable to learn these distributions in a data driven way and embed the mean-field terms into the system model.

The state distribution $\mathbb{P}((z_{t'})_{0 \le t' \le t} \in \prod_{t' \le t} Z_{t'})$ is estimated by a multi-frame learning procedure, which can be performed online or offline. In the offline version, we collect K previous frames as training data. In each frame, the vehicles are annotated with bounding boxes. By compare the positions in two adjacent frames, we can calculate the velocity. Thus, we have the vehicles' positions, velocities and sizes in all frames. Then we make a 2D histogram to estimate the spatially distinct prior distribution of the state variables $(px_t, py_t, sx_t, sy_t, vx_t, vy_t)$. The vehicle position prior distribution is shown in Figure 4b, and the velocity

and size distributions are shown in Figure 6.



(a) sx (b) sy (c) vx (d) vy (e) vx distribution (f) vy distribution

Fig. 6: Mean-field term of velocity and vehicle size. (a)(b) Velocity displayed by pixel intensity. White is positive and black is negative. (c)(d) Vehicle size displayed by pixel intensity. Brightness means larger size, and black area means no vehicle detected.

In Figure 6e and 6f we can see that the speed at position 2,4 is smaller than position 1,3, and vx at position 3,4 is positive and vx at position 1,2 is negative, which is consistent with the main road directions. Thus, we get the transition kernel $\mathcal{K}_t(z_t, \mathcal{L}_{z_t}; z_{t+1})$ for the system model in equation (8), where $\mathcal{L}_{z_t} = \mathbb{P}oz_t^{-1}$. The mean-field terms E_{sx}, E_{sy}, E_{vx}, E_{vy} can be calculated by taking the expected value of the estimated prior distribution. In the online version, we maintain a dynamic mean-field term with the state variables in previous frames:

$$z_{t+1} = (1-\zeta)E[z_t] + \zeta z_t \tag{9}$$

where the learning rate ζ is set to 0.001.

C. Tracking

The tracking procedure starts from the detection result, and the state space is split into independent zones based on the blobs shown in Figure 5d. For each zone, we put N=10 particles, and the initial state parameters (px_0, py_0, sx_0, sy_0) are set as the blob's enveloping rectangle. Since the vehicle's velocity is unknown at the beginning, we set vx_0, vy_0 as the mean-field value $E_{vx}(px_0, py_0), E_{vy}(px_0, py_0)$. Then we add Gaussian noise to shift the particles and let them evolve based on the system dynamic in equation 8. The observation likelihood $l_t(z^i_{s,t}, \mathcal{L}_{z^i_{s,t}}; y_{s,t})$ indicates the confidence of a particle hypothesis, which is measured by the similarity with a given target model. For each particle, we extract the pixel intensity vector from a rectangular tile defined by (px_t, py_t, sx_t, sy_t). The similarity between two tiles p, q is computed by the normalized cross correlation (NCC):

$$s(p,q) = 0.5 * (NCC(p,q) + 1) \tag{10}$$

Aside from the appearance similarity, the confidence of a particle hypothesis also depends on the bias to the expected particle state:

$$d^2(z^i_t, E[z^i_t]) = (z^i_t - E[z^i_t])^2 \tag{11}$$

Therefore, the particle weight before normalization is calculated by:

$$w^i_t = s(p,q)exp(-d^2(z^i_t, E[z^i_t])) \tag{12}$$

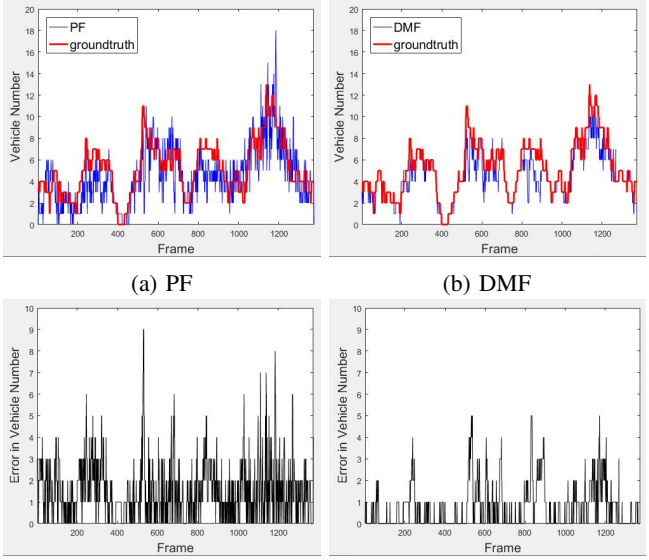
In each frame, the output vehicle locations are obtained from the particles with the highest weight, and the corresponding rectangular tiles are stored as the target appearance model to be used later when it is compared to the other tiles of the particles in the next frame.

For multi-vehicle tracking, we create new tracks for each unassigned zone and store its appearance model. If it is re-detected in the next frame, i.e. the best particle's confidence exceeds some threshold, we increase its age by 1. The track will be accepted when $age \geq 3$. On the other hand, if a track has not been detected for too many consecutive frames, it will be deleted, and the particles deployed in that zone will be removed. The deletion happens when the vehicle goes out of view or to eliminate the noisy detections.

The performance is evaluated by the successful tracking rate. We define the overlap rate of two bounding boxes as the ratio between the intersection and union area. For each vehicle, the tracking in frame t is successful if the overlap rate between the output hypothesis and the ground truth is above 0.5. We compare our approach with particle filter [22] modified by [13]. The particle number is set to be N = 1000, while in our algorithm we assign N = 10 particles in each zone. The processing speed is around 0.6 frame per second, with MATLAB implementation on a PC with 3.50-GHz Intel Core E5-1650 CPU. Most computational load is in foreground detection. Our algorithm can be easily accelerated by parallel computing. Figure 7 shows the performance comparison in terms of the successful tracked vehicle number in all 1376 test frames. Experimental results demonstrate the advantage of our approach. More tracking results are displayed in Figure 8.

IV. Conclusion

This paper proposed a new distributed mean-field type filtering framework for vehicle tracking on highways. The filter has four components: sampling, prediction, decomposition and correction. By decomposition of the entire state space,



(a) PF (b) DMF

(c) Error of PF (d) Error of DMF

Fig. 7: Performance comparison.

Fig. 8: Tracking result in 12 views.

the distributed filters perform locally and focus on their simplified subspace. Based on the foreground detection result, particles are initially deployed near some good hypotheses, which is better than random guessing. A multi-frame learning procedure is added before the tracking task to estimate the prior state distribution. A mean-field term is combined in the system dynamic so that the state prediction depends on not only the previous state but also the statistics of the process. The mean-field filter performs well with small ensembles in the vehicle tracking task.

REFERENCES

[1] J. Zhang, F. Y. Wang, K. Wang, W. H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE ITS Transactions*, vol. 12, no. 4, pp. 1624–1639, Dec 2011.

[2] N. Buch, S. A. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE ITS Transactions*, vol. 12, no. 3, pp. 920–939, Sept 2011.

[3] G. Welch and G. Bishop, "An introduction to the kalman filter," Chapel Hill, NC, USA, Tech. Rep., 1995.

[4] H. Tembine, R. Tempone, and P. Vilanova, "Mean-field learning: a survey," *CoRR*, vol. abs/1210.4657, 2012.

[5] T. Yang, P. G. Mehta, and S. P. Meyn, "A mean field control-oriented approach to particle filtering," *In Proceedings of American Control Conference (ACC)*, pp. 2037–2043, 2011.

[6] B. Djehiche and H. Tembine, *Risk-Sensitive Mean-Field-Type Control Under Partial Observation*. Springer International Publishing, 2016, pp. 243–263.

[7] A. Bensoussan, B. Djehiche, H. Tembine, and P. Yam, "Risk-sensitive mean-field-type control," 2017.

[8] H. Tembine, "Distributed strategic learning for wireless engineers."

[9] J. Gao and H. Tembine, "A mean-field filter for two-step ahead forward-looking problems," July 2016.

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, Sept 2010.

[11] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, July 2012.

[12] J. Gao and H. Tembine, "Distributed mean-field-type filters for big data assimilation," in *2nd IEEE International Conference on Data Science and Systems, Sydney, Australia*, 2016, pp. 1446–1453.

[13] G. Shabat, Y. Shmueli, A. Bermanis, and A. Averbuch, "Accelerating particle filter using randomized multiscale and fast multipole type methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 7, pp. 1396–1407, July 2015.

[14] R. Hostettler and P. M. Djuri, "Vehicle tracking based on fusion of magnetometer and accelerometer sensor measurements with particle filtering," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 4917–4928, Nov 2015.

[15] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review*, vol. 1112, pp. 31 – 66, 2014.

[16] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4 – 21, 2014.

[17] S. Brutzer, B. Hferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *Computer Vision and Pattern Recognition (CVPR)*, June 2011, pp. 1937–1944.

[18] K. Wang, Y. Liu, C. Gou, and F. Y. Wang, "A multi-view learning approach to foreground detection for traffic surveillance applications," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4144–4158, June 2016.

[19] T. Horprasert, D. Harwood, and L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," 1999, pp. 1–19.

[20] A. T. Ihler, J. W. Fischer III, and A. S. Willsky, "Loopy belief propagation: Convergence and effects of message errors," *J. Mach. Learn. Res.*, vol. 6, pp. 905–936, Dec. 2005.

[21] S. Z. Li, *Markov Random Field Modeling in Image Analysis*, 3rd ed. Springer Publishing Company, Incorporated, 2009.

[22] C. Hue, J. P. L. Cadre, and P. Perez, "Tracking multiple objects with particle filtering," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 791–812, Jul 2002.