A Distributed Solution for Multi-Object Tracking and Classification

Samir Hachour*, François Delmotte† and David Mercier†
Univ. Lille Nord de France, UArtois, EA 3926 LGI2A, Béthune, France
Emails:*samir_hachour@ens.univ-artois.fr
†firstname.lastname@univ-artois.fr

***Abstract*—This paper presents a distributed solution for multi-object tracking and classification. The state of objects is partially observed by a set of sensors organized in a network. The idea is to exchange partial data throughout the network and provide a complete information at each sensor level. The proposed solution involves a finite time average consensus where existing solutions are based on asymptotic consensus. The consensus algorithm intervenes in both distributed tracking and classification of multiple objects. It is firstly used to complete information about objects trajectories and secondly to complete beliefs concerning the classification. Simulation results show the relevance of the proposed solution.**

I. Introduction

Cars in a highway can be seen as a complex parameter distributed system. To get traffic information, a given number of sensors can be deployed. They can centralize their information to a fusion center and then proceed to an estimation, surveillance or control process. This solution would be very expensive and maybe not feasible. This is due to the wide spatial distribution of the system and the difficulties associated with the connection of sensors to a fusion center. A full connected sensor network may be considered, but this would be more complex than the centralized solution as far as each sensor become a fusion center. This refers to decentralized approaches, an example of a decentralized approach applied to objects tracking can be found in [1].

To deal with realistic constraints like sensors computation limits and limited communication ranges. Solutions avoiding full connection in networks are required. These solutions are referred to as distributed approaches [2]–[5]. Sensor networks in this kind of approaches are seen as communication graphs where sensors represent nodes and communication links represent edges. A node does not need to be connected to all graphs nodes. Data are exchanged only between neighboring nodes and thus completed at each node level. In a distributed estimation context, the Distributed Kalman Filter (DKF) was designed [6], [7]. In related references, DKF is based on a standard average consensus algorithm [8]–[10]. The average consensus algorithm assumes that communication graph is known by each node. Based on this knowledge and data communicated by neighboring nodes, each node is able to converge to the average value of the received data. These aggregated data are used to update DKF. A multi-object tracking solution based on DKFs is proposed in [11]. In this solution, the assignment of observations to known objects is ensured be a Joint Probability Data Association (JPDA) algorithm.

An equivalent tracking solution is proposed in this paper. It is based on DKFs but it uses a finite time average consensus algorithm instead of the standard asymptotic one. This finite time consensus algorithm allows the calculation of an exact average of entering data in a finite time. It can be understood that the finite time consensus based on DKF performance would be equivalent to a centralized Kalman filter performance, since it calculates the exact average value of local data. Farther information about finite time consensus algorithm can be found in [12], [13]. Observations to tracks and tracks to tracks assignments in the proposed solution is performed by a Global Nearest Neighbor (GNN) algorithm which gives an optimal deterministic matching. Of course, assignment step can be replaced by more elaborated probabilistic methods like JPDA or Multiple Hypotheses Tracking (MHT) algorithms, or even, belief functions based assignment methods like [14]. Notice that DKFs, in the proposed solution are incorporated in Interacting Multiple Model (IMM) algorithms in order to handle objects maneuvers. More information about IMMs can be found in [15], [16]. Confirmation of appearing objects and deletion of non-detected ones are ensure by score functions (statistical test) [16], [17].

In addition to the tracking task, a distributed classification strategy is proposed. In [18], a method to propagate beliefs throughout a sensor network is proposed. It is based on a standard asymptotic consensus algorithm. In this paper, a finite time consensus algorithm is used and belief propagation strategy is applied on cars behaviors recognition.

Section II of this paper describes the system of interest. Section III provides some details on the adopted distributed tracking and classification strategies. Section IV highlights the relevance of the proposed solution throughout simulation results and Section V concludes the paper.

II. System description

Cars in a highway are considered as multiple moving objects. Their state vectors evolve as follows:

$$x(k+1) = F_f x(k) + w(k+1), \tag{1}$$

where $x(k)$ represents the state vector at time step k, F_f a linear state matrix, with $f \in \{1,\ 2,\ ...,\ nf\}$ and where nf represents the number of linear possible evolution modes. The vector w represents the state noise considered as Gaussian with covariance matrix Q.

Observations taken by a given sensor $i \in \{1, 2, ..., N\}$ are

modeled as follows:

$$z_i(k) = H_i x(k) + v_i(k), \tag{2}$$

where H_i represents the observation matrix and v_i is the sensor noise considered as Gaussian with covariance matrix R_i.

It is supposed that the state vector of each object is partially and complementary observed by the sensors. The global observation model would be:

$$z(k) = [z_1^{'}(k)\ z_2^{'}(k)\ ...\ z_N^{'}(k)]^{'}, \tag{3}$$

equivalently, the global sensor noise vector is given by:

$$v(k) = [v_1^{'}(k)\ v_2^{'}(k)\ ...\ v_N^{'}(k)]^{'}, \tag{4}$$

the global covariance matrix is given as follows:

$$R = [R_1\ R_2\ ...\ R_N], \tag{5}$$

and finally, the equivalent global observation matrix is given by:

$$H = [H_1^{'}\ H_2^{'}\ ...\ H_N^{'}]^{'}, \tag{6}$$

This hypothesis on the observation model allows us to make the following equivalences [6]:

$$\begin{cases} H^{'} R^{-1} H = \sum_{i=1}^{N} H_i^{'} R_i^{-1} H_i \\ H^{'} R^{-1} z(k) = \sum_{i=1}^{N} H_i^{'} R_i^{-1} z_i(k) \end{cases} \tag{7}$$

Equivalences in (7) allows the Kalman filters based distributed tracking to be performed. This is explained in the following section.

III. Distributed tracking and classification strategy

A. Distributed tracking solution

The adopted distributed tracking solution is illustrated in Figure 1. The main steps of the algorithm are described in the following.

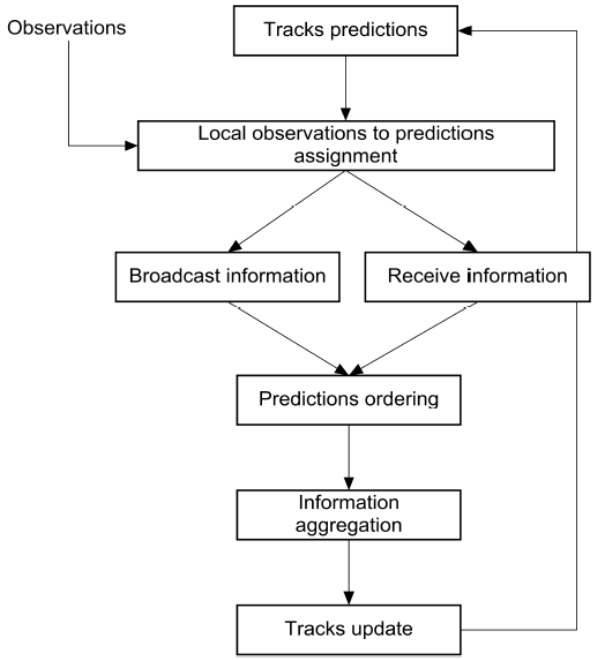

Fig. 1. Flowchart of the proposed distributed tracking solution

1) Tracks predictions: The prediction step of the distributed tracking algorithm is depicted in Algorithm 1.

Algorithm 1 Tracks predictions.

Require: A priori estimates: $\hat{x}_{i,f}^t(k-1|k-1)$, $\hat{S}_{i,f}^t(k-1|k-1)$, $t = \{1, 2, ..., n_i\}$, $i = \{1, 2, ..., N\}$, $f = \{1, 2, ..., nf\}$. Models parameters: F_f, R, Q and models probabilities $\sigma_f(k-1)$ calculated in IMMs process.

Ensure: State $\bar{x}_i^t(k|k-1)$ and covariance matrix $\bar{S}_i^t(k|k-1)$. For each Kalman filter $f = \{1, 2, ..., nf\}$ in the IMM:

1 State vector $\hat{x}_{i,f}^t(0|0)$ and covariance matrix $\hat{S}_{i,f}^t(0|0)$ initialization, $k = 0$.

2 State vector $\bar{x}_{i,f}^t(k|k-1)$ and covariance matrix $\bar{S}_{i,f}^t(k|k-1)$:

$$\bar{x}_{i,f}^t(k|k-1) = F_f \hat{x}_{i,f}^t(k-1|k-1). \tag{8}$$

$$\bar{S}_{i,f}^t(k|k-1) = F_f \hat{S}_{i,f}^t(k-1|k-1) F_f^{'} + Q. \tag{9}$$

Global quantities: $\bar{x}_i^t(k|k-1)$, $\bar{S}_i^t(k|k-1)$.

$$\bar{x}_i^t(k|k-1) = \sum_{f=1}^{nf} \sigma_f^t \bar{x}_{i,f}^t(k|k-1). \tag{10}$$

$$\bar{S}_i^t(k|k-1) = \sum_{f=1}^{nf} \sigma_f^t \bar{S}_{i,f}^t(k|k-1). \tag{11}$$

Notice that, notation $\bar{x}_{i,f}^t(k|k-1)$ refers to the state vector prediction of target t, made by filter f, at node i, where $i = \{1, 2, ..., N\}$, $f = \{1, 2, ..., nf\}$, $t = \{1, 2, ..., n_i\}$, with N being the number of nodes in the network, nf the number of Kalman filters in an IMM and n_i the number of targets known by node i.

2) Observations to tracks assignment: Once tracks predictions have been completed at each sensor (node) i, they are compared to some detected observations z_i^j, with $j = 1, 2, ..., m_i$ where m_i refers to the number of observations detected by sensor i.

Assignment of observations to tracks is done by means of a Global Nearest Neighbor (GNN) algorithm. It is an optimal solution which minimizes the global distance between observations and predictions. Distances are calculated as follows:

$$d_{t,j}(k) = (z_i^j(k) - H_i^t\ \bar{x}_i^t(k|k-1))^{'}(S_i^t)^{-1} \\ (z_i^j(k) - H_i^t\ \bar{x}_i^t(k|k-1)), \tag{12}$$

where, $i = \{1, 2, ..., N\}$, $t = \{1, 2, ..., n_i\}$ and $j = \{1, 2, ..., m_i\}$. Assignment matrix, formed by distances $[d_{t,j}]$ is resolved with Munkres algorithm [19].

After observations to objects assignment, each node i is able to provide the following information:

- Set of predictions: $\bar{X}_i(k|k-1) = \{\bar{x}_i^1(k|k-1), \bar{x}_i^2(k|k-1), ..., \bar{x}_i^{n_i}(k|k-1)\}$.
- Detection index: $r_i(k) = \{r_i^1(k), r_i^2(k), ..., r_i^{n_i}(k)\}$, where, $r_i^t(k) = 1$ if node i detects object t at time step k and it equals zero otherwise.
- Observation information $\nu_i(k) = \{\nu_i^1(k), \nu_i^2(k), ..., \nu_i^{n_i}(k)\}$, $\mu_i(k) = \{\mu_i^1(k), \mu_i^2(k), ..., \mu_i^{n_i}(k)\}$:

$$\nu_i^t(k) = r_i^t(k)(H_i^{'}R_i^{-1}H_i), \tag{13}$$

$$\mu_i^t(k) = r_i^t(k)(H_i^{'}R_i^{-1}z_i(k)). \tag{14}$$

- Local likelihoods: $\lambda_i(k) = \{\lambda_i^1(k), \lambda_i^2(k), ..., \lambda_i^{n_i}(k)\}$. We suppose that each object t can be classified among a set of nc classes $\{c_1,\ c_2,\ ...,\ c_{nc}\}$. Therefore, $\lambda_i^1(k)$ represents a vector containing all classes likelihoods. For instance, $\lambda_i^t(z_i|c_j)$ represents the likelihood that object t belongs to class c_j based on local observation z_i made by node i. Notice that classification in this work is based on objects kinematic data, thus, classes likelihoods are simply mixtures of Kalman filters likelihoods in the IMMs.

To summarize: each node i has to send a message $msg_i = \{\bar{X}_i(k|k-1), r_i(k), \nu_i(k), \mu_i(k), \lambda_i(k)\}$ in the communication step illustrated in Figure 1. At the same time, the same message is received by the neighboring nodes in the communication graph.

Now the question is: how to process these data in order to complete the information at each node level?

First, each node starts by a predictions ordering step. This step needs to save sets of predictions in nodes memories. After that, another assignment algorithm is processed, it is called tracks to tracks assignment.

3) Tracks to tracks assignment: This step allows nodes to get a common order of the tracks. An optimal assignment step is performed, for each couple of nodes. Weight of assigning track t of node i to track l of node j, for example, is calculated as follows:

$$D_{t,l}(k) = (\bar{x}_i^t(k|k-1) - \bar{x}_j^l(k|k-1))^{'}(\bar{S}_i^t(k|k-1) \\ +\bar{S}_j^l(k|k-1))^{-1}(\bar{x}_i^t(k|k-1) - \bar{x}_j^l(k|k-1)), \tag{15}$$

As in the former case, assignment matrix formed by distances $[D_{t,l}]$, with $t = \{1, 2, ..., n_i\}$ and $l = \{1, 2, ..., n_j\}$, is also resolved using Munkres algorithm.

As illustrated in Figure 1, after predictions ordering step, comes the data aggregation step. Quantities $\nu_i(k)$, $\mu_i(k)$ and $\lambda_i(k)$, with $i = \{1, 2, ..., N\}$ do not have to be saved in nodes memories, they are processed by an average finite time consensus algorithm. Likelihoods $\lambda_i(k)$ concerns objects classification, their aggregation process is described in Section III-B.

a) Average consensus algorithm: The average consensus algorithms aim to calculate the average value of some entering data as illustrated by the following equation:

$$ave(y(0)) = \frac{1}{N}\sum_{i=1}^{N} y_i(0). \tag{16}$$

The average "ave" in Equation 16 can be calculated in a distributed way according to the following dynamic equation:

$$y(t+1) = Py(t), \tag{17}$$

where $y(t) = [y_1(t)\ y_2(t)\ \dots\ y_N(t)]^{'}$ and P is a double stochastic $(N \times N)$ matrix ($P_{i,j} \geq 0$, $\sum_{i=1}^{N} P_{i,j} = 1$, $\sum_{j=1}^{N} P_{i,j} = 1$,). Matrix P is compatible with the communication graph ($P_{i,j} > 0$ if node i is linked with node j, $P_{i,j} = 0$ otherwise). Matrix P is constant when the communication graph is fix and satisfies the following condition in asymptotic average consensus context [7], [8]:

$$\lim_{t\to\infty} P = \frac{1}{N}\mathbf{11}, \tag{18}$$

where $\mathbf{11}$ represents an $(N \times N)$ unit matrix. Methods to parameterize matrix P can be found in [10].

In the finite time average consensus algorithm presented in [12], [13], instead of using matrix P in Equation (17), a product of a set of graph compatible matrices $Q(t)$ is used:

$$\prod_{t=1}^{L} Q(t) = \frac{1}{N}\mathbf{11}, \tag{19}$$

An average value is reached in L iterations. The parameter L represents the number of distinct non-zero eigenvalues of the communication graph Laplacian matrix. Broader information on the calculation of matrices $Q(t)$ can be found in [12], [13].

4) Information aggregation: As explained by Equations in (7), the global quantities required for tracking is a sum of local quantities. Using the consensus algorithm, the following sums can be calculated:

$$\nu^t(k) = Nb^t ave(\{\nu_1^t(k)\}_{i=1}^{Nb^t}), \tag{20}$$

where $Nb^t = \sum_{i=1}^{N} r_i^t(k)$ represents the number of sensors which have detected object t at time k. The parameters $\mu_i^t(k)$ can be aggregated in the same manner throughout a second consensus algorithm which can be run in parallel:

$$\mu^t(k) = Nb^t ave(\{\mu_1^t(k)\}_{i=1}^{Nb^t}), \tag{21}$$

Using the calculated global quantities, the objects states estimation of time step k can be complete. This is done through the update step of Kalman filters in the corresponding IMMs.

5) Tracks updates: Once global quantities are provided, Kalman filters can be updated according to the Algorithm in 2.

Algorithm 2 Tracks updates.

Require: Predictions $\bar{x}_{i,f}^t(k|k-1)$, $\bar{S}_{i,f}^t(k|k-1)$ calculated in Algorithm 1.
Aggregated quantities: $\mu^t(k)$, $\nu^t(k)$.
Kalman filters probabilities $\mu_f^t(k)$ (calculated in IMMs process).

Ensure: State $\hat{x}_i^t(k|k)$ and covariance matrix $\hat{S}_i^t(k|k)$.
For each Kalman filter $f = \{1, 2, ..., nf\}$ in the IMM:

1 Covariance matrix update:

$$(\hat{S}_{i,f}^t(k|k))^{-1} = (\bar{S}_{i,f}^t(k|k-1))^{-1} + \nu^t(k). \tag{22}$$

2 State vector $\hat{x}_{i,f}^t(k|k)$ update:

$$\hat{x}_{i,f}^t(k|k) = \hat{S}_{i,f}^t(k|k)[(\bar{S}_{i,f}^t(k|k-1))^{-1}\bar{x}_{i,f}^t(k|k-1) + \mu^t(k)]. \tag{23}$$

Global quantities (external use): $\hat{x}_i^t(k|k)$, $\hat{S}_i^t(k|k)$.

$$\hat{x}_i^t(k|k) = \sum_{f=1}^{nf} \sigma_f^t(k)\hat{x}_{i,f}^t(k|k). \tag{24}$$

$$\hat{S}_i^t(k|k) = \sum_{f=1}^{nf} \sigma_f^t(k)\hat{S}_{i,f}^t(k|k). \tag{25}$$

Updating Kalman filters at nodes level with global aggregated quantities allows information concerning objects states to be completed even if the observability of sensors (nodes) is partial. Of course, the estimation quality would be equivalent to an estimation done by a fusion center, except that, the distributed approach offers the scalability, short range communication and faster processing advantages.

B. Distributed classification

A third consensus algorithm can be run, it aims to aggregate likelihoods concerning objects classification. The question is: how to formulate a global likelihood as a function of an average value of local likelihoods?

It is mentioned in [18], that posterior probabilities of a given number of classes $c_j \in \{c_1, c_2, ..., c_{nc}\}$, where nc is the number of classes, can be calculated as follows:

$$p_k(c_j|z(k)) = \alpha p_{k-1}(c_j) \prod_{i=1}^{N} \lambda(z_i(k)|c_j), \tag{26}$$

where α is a normalizing factor.

Here, the interest is on a distributed calculation of the global likelihood noted W:

$$W = \prod_{i=1}^{N} \lambda(z_i(k)|c_j), \tag{27}$$

which can easily be expressed as an average value of local quantities:

$$W' = \frac{1}{N} log(W) = \frac{1}{N} \sum_{i=1}^{N} log(\lambda(z_i(k)|c_j)). \tag{28}$$

The quantity $W' = ave(\{log(\lambda(z_i(k)|c_j))\}_{i=1}^{N})$ can thus be calculated in a distributed manner using consensus algorithm. Once this is done, the global likelihood is deduced: $(W = exp(NW')$.

The global likelihood W can then be used in Equation (26) to update classes posterior probabilities.

C. Discussion

A considerable amount of works had considered the problem of multi-sensor, multi-target tracking [20]–[25]. They essentially studied the problem of senors estimates fusion, which is a non-trivial problem, since data correlations need to be considered in order to avoid data incest in fusion centers.

Let us note that in the method proposed in this paper, sensors estimates are not fused. A fusion of sensors observations, with noises considered as independent, is realized.

IV. Simulation results

A. Description

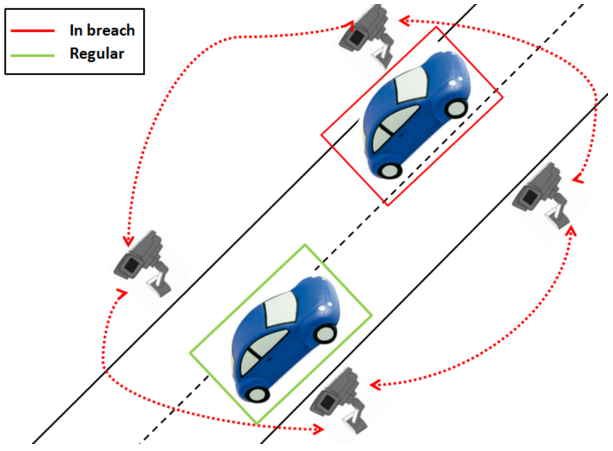

Fig. 2. Simulated system.

Figure 2 illustrates the simulated example. The idea is to track vehicles in a highway using a given number of wireless sensors having partial knowledge. Vehicles evolve according to model (1), where the state matrix is given by:

$$F_f = \begin{bmatrix} 1 & s_f^x \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & s_f^y \Delta T \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where ΔT is the sampling time. Parameters s_f^x and s_f^y represent objects speeds $(meter/second)$ according to x and y directions, respectively. Index $f \in \{1,\ 2,\ ...,\ nf\}$ refers to speed intensity. According to different values of s_f^x and s_f^y, a set $M = \{m_1, m_2, ..., m_6\}$ of $nf = 6$ different models are designed and used by each object IMM to track objects movements. As illustrated in Figure 2, two classes are defined, which are both clusters composed of 3 models.

- Regular behavior class: low velocity models.
- In breach class: high velocity models.

Three sensors are designed to track and classify the objects. The sensors have a partial observability on objects environment. Sensors 1 and 2 take the x direction measures of objects according to the model (2), with:

$$H_1 = H_2 = [\ 1 \quad 0 \quad 0 \quad 0\ ].$$

A third sensor can only observe y direction positions of the objects, it takes measurements according to the model in (2), using the observation matrix:

$$H_3 = [\ 0 \quad 0 \quad 1 \quad 0\ ].$$

B. Results

1) Tracking with & without consensus: The first simulation is about tracking task. As illustrated in Figures 3 and 4, two objects evolves in x and y direction $(y = x)$. Figure 3 shows results of local tracking performed by sensors 1 and 3 (sensor 1 measures objects position according to x direction only and sensor 3 measures objects position according to y direction only).

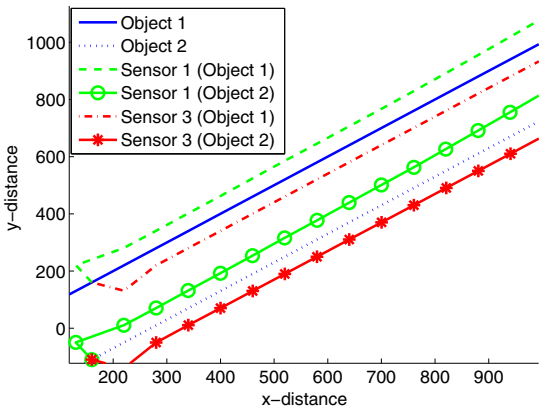

Fig. 3. Trajectories estimation without consensus.

Notice that estimation of sensor 2 is equivalent to the estimation of sensor 1 since they have the same observation models. It can be seen that sensors fail to estimate object trajectories with their local observation.

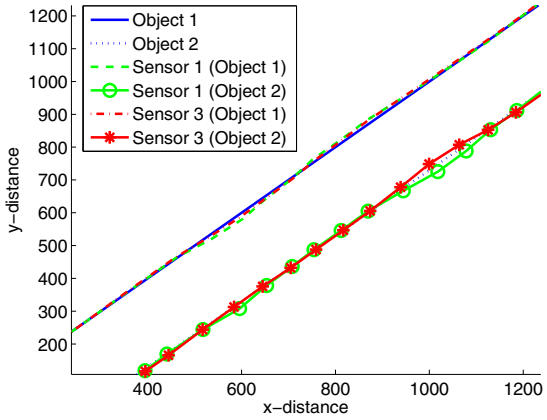

Fig. 4. Trajectories estimation with consensus.

Figure 4 gives objects trajectories estimations results. These results show that sensors succeed in estimating objects trajectories using the presented distributed estimation strategy.

2) Classification with & without consensus: In this section, it is assumed that all sensors make a correct estimation of objects trajectories (using the distributed strategy). The focus here is on the classification task, where classification results are based on sensors likelihoods (with and without consensus aggregation). Classification aims to track the behavior of vehicles (vehicles with regular behavior, vehicles in breach).

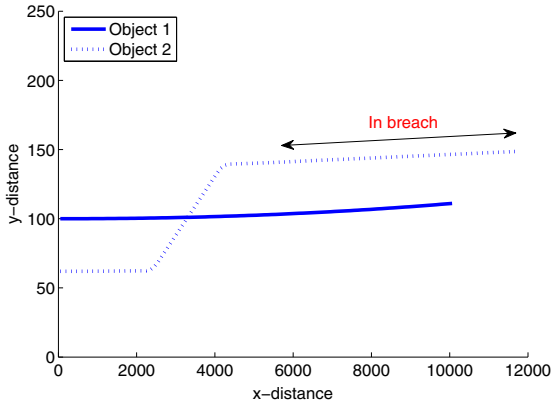

Fig. 5. Real trajectories of two vehicles on the road.

Figure 5 describes the simulated scenario which is about two vehicles evolving on a highway. Interest is on the classification of object 2 which adopt a regular behavior at the beginning of its evolution and pass to a non-regular behavior by exceeding the authorized speed (in x direction), as illustrated in Figure 5.

Local classification (without consensus) results provided by sensors 1 and 2 are given in Figure 6. It can be seen that classification results are as expected: regular and then in breach behavior.

It can be seen in Figure 7 that unlike sensors 1 and 2, sensor 3 does not succeed to correctly classify object 2. This because sensor 3 knowledge is based on y direction of the road and

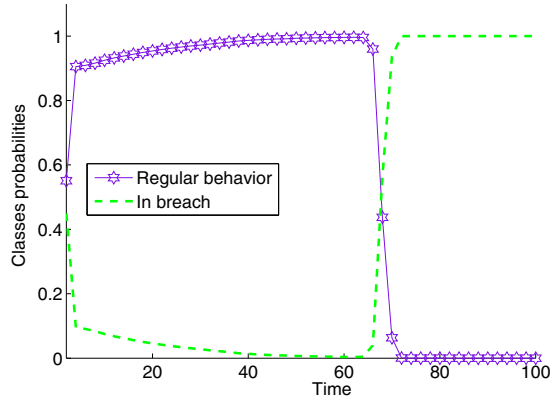

Fig. 6. Sensors 1 and 2 classification result (without consensus).

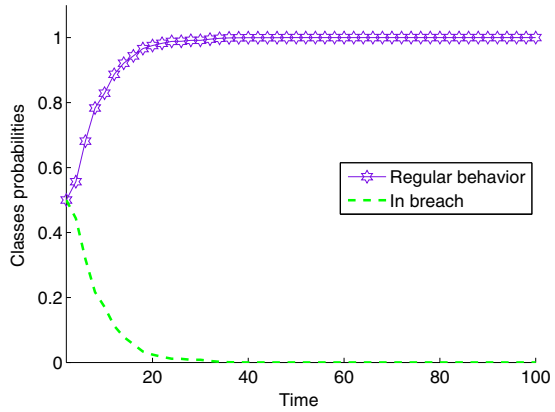

Fig. 7. Sensor 3 classification result (without consensus).

that maneuver of object 2 is done in x direction. This would be corrected by the presented data aggregation strategy.

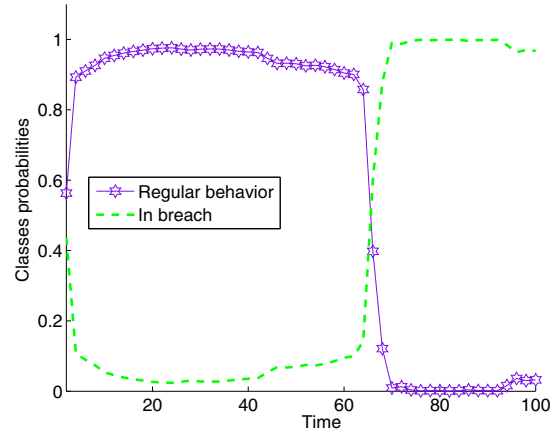

Fig. 8. Sensors classification result (with consensus).

Using the distributed classification strategy, it can be seen that all sensors succeed in performing a correct classification of object 2. Classification result obtained by the three sensors is depicted in Figure 8

V. Conclusion

This paper proposes a complete solution for distributed object tracking and classification. The multi-object tracking is ensured by distributed-Kalman-filter-based IMMs. The use of IMMs allows the maneuvers of objects to be handled. The distributed Kalman filters are based on finite time consensus algorithms. Consensus algorithms allow local and partial data of multiple sensors to be aggregated and used for objects state estimation update. Therefore, this avoid the complexity of centralized solutions. Observations to tracks and tracks-to-tracks assignments are ensured by GNN algorithms. Objects appearances and disappearances are managed by score functions [16], [17].

In addition to the tracking task, a distributed classification method is also provided. It is a simple classification example based on kinematic data. The idea consists on a global and complete classes likelihoods calculations based on local and partial ones. Global likelihoods calculations are based on a distributed finite time consensus algorithm. They are used to calculate classes posterior probabilities and allow to get a complete classification at each node level.

Farther, consensus algorithms with dynamical graphs can be considered. This would give an appreciable solution concerning information sharing in systems like vehicles traffic. Vehicles themselves would belong to graph sensors.

References

[1] B-S. Rao and H-F. Durrant-Whyte. Fully decentralised algorithm for multisensor kalman filtering. In *Control Theory and Applications*, volume 138, pages 413–420. IET, 1991.

[2] H. Li, F. Xiao, J. Zhou, and X-R. Li. Nonlinear distributed estimation fusion that reduces mean square error. In *Proceedings of the 16th International Conference on Information Fusion*, 2013.

[3] A. Oka and L. Lampe. Distributed target tracking using signal strength measurements by a wireless sensor network. *IEEE Journal on Selected Areas in Communications*, 28(7):1006–1015, 2010.

[4] L. Chen, M. Cetin, and A-S. Willsky. Distributed data association for multi-target tracking in sensor networks. 2005.

[5] D-W. Casbeer and R. Beard. Distributed information filtering using consensus filters. In *American Control Conference*, pages 1882–1887. IEEE, 2009.

[6] R. Olfati-Saber. Distributed Kalman filter with embedded consensus filters. In *44th IEEE Conference on Decision and Control*, pages 8179–8184. IEEE, 2005.

[7] R. Olfati-Saber. Distributed kalman filtering for sensor networks. In *46th IEEE Conference on Decision and Control*, pages 5492–5498. IEEE, 2007.

[8] F. Garin and L. Schenato. A survey on distributed estimation and control applications using linear consensus algorithms. In *Networked Control Systems*, pages 75–107. Springer, 2010.

[9] A. Olshevsky and J-N. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM review*, 53(4):747–772, 2011.

[10] S. Del Favero. *Analysis and development of consensus-based estimation schemes*. PhD thesis, degli Studi di Padova, 2010.

[11] N-F. Sandell and R. Olfati-Saber. Distributed data association for multi-target tracking in sensor networks. In *47th IEEE Conference on Decision and Control.*, pages 1085–1090. IEEE, 2008.

[12] A-Y. Kibangou. Finite-time average consensus based protocol for distributed estimation over awgn channels. In *50th IEEE Conference on Decision and Control and European Control Conference*, pages 5595–5600. IEEE, 2011.

[13] A-Y. Kibangou. Graph laplacian based matrix design for finite-time distributed average consensus. In *American Control Conference (ACC)*, pages 1901–1906. IEEE, 2012.

[14] N. El Zoghby, V. Cherfaoui, and T. Denoeux. Optimal object association from pairwise evidential mass functions. In *Proceedings of the 16th International Conference on Information Fusion*, 2013.

[15] Y. Bar-Shalom, X.R. Li, T. Kirubarajan, and J. Wiley. *Estimation with applications to tracking and navigation*. Wiley Library, 2001.

[16] S.S. Blackman and R. Popoli. *Design and analysis of modern tracking systems*. Artech House, Norwood, MA, USA, 1999.

[17] S. Hachour, F. Delmotte, D. Mercier, and E. Lefèvre. Object tracking and credal classification with kinematic data in a multi-target context. *Information Fusion*. In press.

[18] R. Olfati-Saber, E. Franco, E. Frazzoli, and J-S. Shamma. Belief consensus and distributed hypothesis testing in sensor networks. In *Networked Embedded Sensing and Control*, pages 169–182. Springer, 2006.

[19] F. Bourgeois and J-C. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802–804, 1971.

[20] Y. Bar-Shalom. Multitarget-multisensor tracking: advanced applications. *Norwood, MA, Artech House*, 1, 1990.

[21] A.M.A Aziz. New data fusion algorithms for distributed multi-sensor multi-target environments. Technical report, September 1999.

[22] H-B. Mitchell. *Data Fusion: Concepts and Ideas*. Springer, 2012.

[23] M. Liggins, D. Hall, and J. Llinas. *Handbook of multisensor data fusion: theory and practice*. CRC press, 2008.

[24] H. Durrant-Whyte and T.C. Henderson. Multisensor data fusion. *Springer Handbook of Robotics*, pages 585–610, 2008.

[25] S. Fabre, A. Appriou, and X. Briottet. Presentation and description of two classification methods using data fusion based on sensor management. *Information Fusion*, 2(1):49–71, 2001.