



Fig. 2. The Cycab Platform.

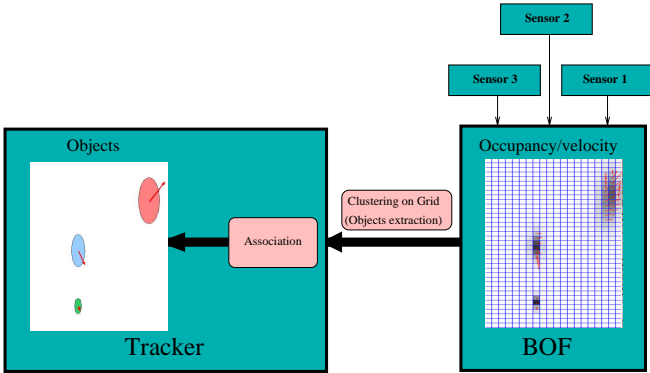


Fig. 3. Sensing/Tracking system architecture.

- 2) using both occupancy and velocity estimates in order to better separate the extracted clusters.

Thanks to these two ideas, the observation-to-track association becomes implicit in the majority of situations. Indeed, a re-clustering and explicit association is needed only when ambiguous associations occur. The computational cost of this approach is linear to the number of dynamic objects detected, so as to be suitable for scenes in cluttered environment.

Our approach has been tested on the real data collected on real cars in highway and cluttered urban environments (Fig. 1), and also on our Cycab experimental platform (Fig. 2).

The paper is organized as follows. In the next section, the ‘‘Bayesian Occupancy Filter’’ (BOF) algorithm is briefly described. The ‘‘Fast Clustering-Tracking’’ algorithm is presented in Section III. In Section IV, the experimental results of our approach on the real data collected on the Cycab platform are provided. Finally, conclusions are drawn in Section V.

II. BAYESIAN OCCUPANCY FILTER (BOF)

The Bayesian Occupancy Filter (BOF) is represented as a two-dimensional grid-based decomposition of the environment. Each cell of the grid contains two probability distributions: (i) the probability distribution over the occupancy of the cell (ii) and the probability distribution over its velocity. Given a set of input sensor readings, the BOF algorithm allows to update the occupancy/velocity estimates of each grid cell.

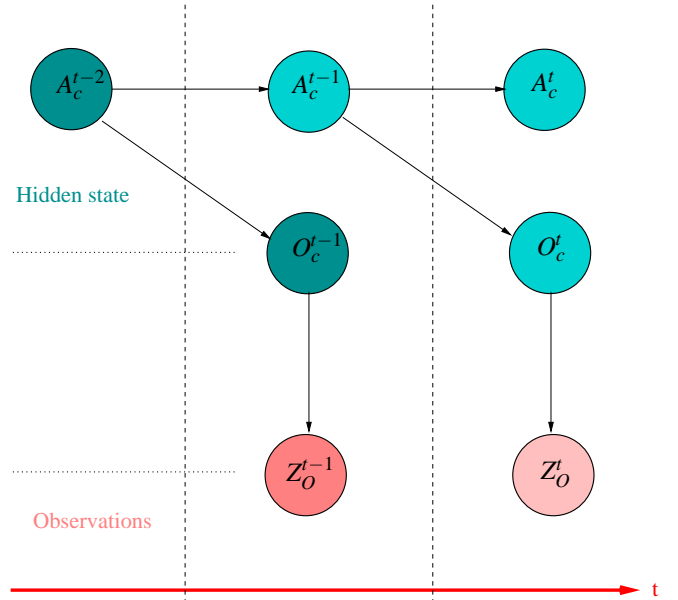


Fig. 4. The Dynamic Bayesian Network corresponding to the BOF model. Here, we suppose that only occupancy sensors are available.

BOF is a special implementation of the Bayesian filter approach [6] [5]. This approach addresses the general problem of recursively estimating the posterior probability distribution $P(X^k | Z^k)$ of the state X of a system conditioned on its observation Z . This posterior distribution is obtained in two stages: prediction and estimation. The prediction stage computes an a priori prediction of the target’s current state known as the prior distribution. The estimation stage then computes the posterior distribution by using the prediction with the current measurement of the sensor.

In the case of the BOF, using this prediction/estimation scheme allows filtering out false alarms, miss-detections, and localization errors in sensors data readings. Figure 1 shows an example of BOF output using a computer vision car detector.

The Bayesian model presented in the following text is a reformulation of the one we presented in [1]. The aim of this reformulation is to make clearer the strong link between the discretization of the space and the discretization of the velocity, which reduces the number of the used random variables and makes the model easier to explain. The key idea of the model is to represent the 2D space using a regular grid. Given this space discretization and assuming that objects do not overlap, the velocity of a given c cell at a time t is directly linked to the identity of its antecedent cell A_c from which the content of cell c moved between $t - 1$ and t . In other words, we can define the velocity of a given cell by providing the index of its antecedent. Therefore, estimating the velocity of a given cell is equivalent to estimating a probability table over all its possible antecedents. Possible antecedents of a cell are defined by providing a neighbourhood from which the cell is reachable in a time step. This model applies also to velocities needing more than one time step for a neighbour cell to reach c . However, for simplicity we will assume only

one-step velocities (neighbours reaching c in one time step).

The BOF model is shown graphically in Fig. 4 and is described as follows:

A. Variables

For a given cell having $c \in \mathcal{Y}$ as index in the grid, let:

- $A_c^t \in \mathcal{A}_c \subset \mathcal{Y}$ represents each possible antecedent of cell c over all the cells in the grid domain \mathcal{Y} . The set of antecedent cells of cell c is denoted by \mathcal{A}_c and is defined as a neighbourhood of the cell c .
- $A_c^{t-1} \in \mathcal{A}_c \subset \mathcal{Y}$ the same as A_c^t but for the previous time step.
- $O_c^t \in \mathcal{O} \equiv \{0,1\}$ is a boolean variable representing the state of the cell in terms of occupancy at time t , either $[O_c = 1]$ if occupied, $[O_c = 0]$ if empty. Given the independency hypothesis, the occupancy of each cell at time t is considered apart from the occupancy of its neighbouring cells at time t .
- $Z_i^t \in \mathcal{Z}, 1 \leq i \leq S \in \mathbb{N}$, is a generic notation for measurements yielded by each sensor i , considering a total of S sensors yielding a measurement at the considered time instant.

B. Joint distribution factors

The following expression gives the decomposition of the joint distribution of the relevant variables according to Bayes' rule and dependency assumptions:

$$P(A_c^{t-1} A_c^t O_c^t Z_1^t \dots Z_S^t) = P(A_c^{t-1}) P(A_c^t | A_c^{t-1}) P(O_c^t | A_c^{t-1}) \prod_{i=1}^S P(Z_i^t | A_c^t O_c^t) (1)$$

The parametric form and semantics of each component of the joint decomposition are as follows:

- $P(A_c^{t-1})$ is the probability for a given neighbouring cell A_c to be the antecedent of c at time $t-1$. In order to represent the fact that cell c is *a priori* equally reachable from all possible antecedent cells in the considered neighbourhood, this probability table is initialized as uniform and is update in each time step.
- $P(A_c^t | A_c^{t-1})$ is the distribution over antecedents at time t given the antecedent of cell c at $t-1$. It represents the prediction (dynamic) model over velocity. If we assume a perfect *constant velocity hypothesis* between the two time frames $t-1$ and t , this distribution is simply:

$$P(A_c^t | A_c^{t-1}) = P(A_{A_c^{t-1}}^{t-1}).$$

In other words, the predicted probability is simply the probability at the preceding time instant for the antecedent at $t-1$.

Considering imperfect *constant velocity hypothesis* is possible by introducing the predicate $E \in \{0,1\} \equiv$ "There was an erroneous prediction", and assuming a probability $P(E) = \varepsilon$. This value is a parameter of the system and corresponds of the probability of not respecting the *constant velocity hypothesis*. We have:

$$- P(A_c^t | A_c^{t-1} \neg E) = P(A_{A_c^{t-1}}^{t-1}),$$

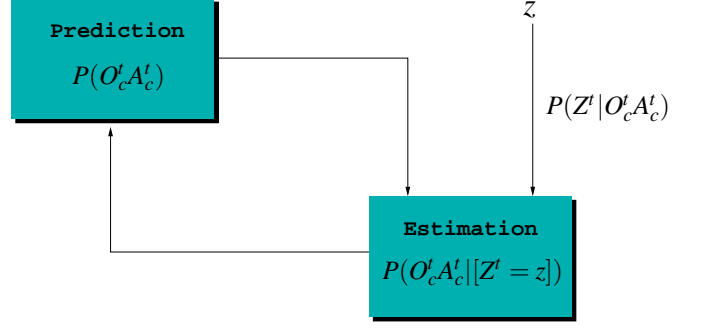


Fig. 5. Bayesian filtering in the estimation of occupancy and velocity distribution in the BOF grids.

$$- P(A_c^t | A_c^{t-1} E) = \mathcal{U}(A_c^t),$$

where $\mathcal{U}(A_c^t)$ denotes a uniform distribution on A_c^t to say that all possible antecedents have the same probability when *constant velocity hypothesis* is not respected. Thus, $P(A_c^t | A_c^{t-1})$ may be written as a mixture:

$$P(A_c^t | A_c^{t-1}) = P(\neg E) P(A_c^t | A_c^{t-1} \neg E) + P(E) P(A_c^t | A_c^{t-1} E).$$

Which leads to:

$$\begin{aligned} P(A_c^t | A_c^{t-1}) &= (1 - \varepsilon) P(A_{A_c^{t-1}}^{t-1}) + \varepsilon \mathcal{U}(A_c^t) \\ &= (1 - \varepsilon) P(A_{A_c^{t-1}}^{t-1}) + \varepsilon / \|\mathcal{A}_c\|, \end{aligned}$$

where $\|\mathcal{A}_c\|$ is the cardinality of the considered antecedents set \mathcal{A}_c .

- $P(O_c^t | A_c^{t-1})$ is the distribution over occupancy given the antecedent of cell c at $t-1$. It represents the prediction (dynamic) model over occupancy. Similarly to $P(A_c^t | A_c^{t-1})$, the term $P(O_c^t | A_c^{t-1})$ may be written as a mixture:

$$\begin{aligned} P(O_c^t | A_c^{t-1}) &= (1 - \varepsilon) P(O_{A_c^{t-1}}^{t-1}) + \varepsilon \mathcal{U}(O_c^t) \\ &= (1 - \varepsilon) P(O_{A_c^{t-1}}^{t-1}) + \varepsilon / 2. \end{aligned}$$

- $P(Z_i^t | A_c^t O_c^t)$ is the *direct model* for sensor i . It yields the probability of a measurement given the occupancy O_c^t and the antecedent (velocity) A_c^t of cell c . Measurements for all sensors are assumed to have been taken *independently from each other*. For sensors providing measurements depending exclusively of occupancy, this distribution can be written as $P(Z_i^t | O_c^t)$. In the same manner, for sensors providing measurements depending exclusively of velocity, this distribution can be written as $P(Z_i^t | A_c^t)$.

C. Occupancy and velocity estimation using the BOF model

At each time step, the estimation of the occupancy and velocity of a cell is answered through Bayesian inference on the model given in Equation (1). This inference leads to a Bayesian filtering process (Fig. 5). In this context, the prediction step propagates cell occupancy and antecedent (velocity) distributions of each cell in the grid to get the prediction $P(O_c^t A_c^t)$. In the estimation step, $P(O_c^t A_c^t)$ is updated by

taking into account the observations yielded by the sensors $\prod_{i=1}^S P(Z_i^t | A_c^t O_c^t)$ to obtain the a posteriori state estimate $P(O_c^t | A_c^t | [Z_1^t \dots Z_S^t])$. This allows, by marginalization, to compute $P(O_c^t | [Z_1^t \dots Z_S^t])$ and $P(A_c^t | [Z_1^t \dots Z_S^t])$ that will be used for prediction in the next iteration.

It's important to notice that the distribution $P(A_c^t)$ over antecedents (velocity) is updated even when no velocity sensors are available. Indeed, suppose we have only one occupancy sensor described by the model $P(Z_O^t | O_c^t)$. The a posteriori distribution $P(A_c^t | [Z_O^t])$ leads to the formula:

$$P(A_c^t | [Z_O^t]) \propto \sum_{A_c^{t-1} \in \mathcal{A}_c} P(A_c^{t-1}) P(A_c^t | A_c^{t-1}) \sum_{O_c^t \in \{0,1\}} P(O_c^t | A_c^{t-1}) P([Z_O^t] | O_c^t). \quad (2)$$

In this case, the update is based exclusively on the occupancy observations.

When an additional velocity sensor $P(Z_V^t | A_c^t)$ is available, it should be used to update the estimate (2) as follows:

$$P(A_c^t | [Z_O^t Z_V^t]) \propto P(A_c^t | [Z_O^t]) P([Z_V^t] | A_c^t).$$

Finally, as the relationship between the velocities and antecedent IDs (indexes) is deterministic (an antecedent id corresponds to a (vx,vy) vector), the probability $P(V_c^t)$ over the velocity is summarised as a 2D Gaussian distribution using the distribution $P(A_c^t)$. Therefore, the output of the BOF algorithm at each time step t is a grid in which each cell c contains (i) an occupancy probability $P(O_c^t)$ and (ii) a 2D Gaussian distribution $P(V_c^t)$ over velocity.

III. THE "FAST CLUSTERING-TRACKING" ALGORITHM

The object-level representation is mandatory for applications needing high-level representations of obstacles and their motion. This needs a robust multi-target tracking system allowing to estimate of the position and the velocity of each moving object.

The main difficulty of multi-target tracking is known as the "data association" problem. It includes observation-to-track association and track management problems. The main goal of observation-to-track association is to decide whether a new sensor observation corresponds to an existing track. Track management includes deciding whether existing tracks should be maintained, deleted, or if new tracks should be created. Numerous methods exist to perform data association. The reader is referred to [7] for a complete review of the existing tracking methods with one or more sensors.

In the BOF framework, we proposed in [1] to use a layered architecture as shown in Fig. 3 to obtain the object-level representation. In [1], the data association is implemented using a classical JPDA algorithm within the proposed architecture.

In cluttered environments with large numbers of moving objects, the JPDA [3] suffers from the combinational explosion of hypotheses. To overcome this problem, we propose the "Fast Clustering-Tracking" algorithm. This algorithm could be roughly divided into a clustering module,

an ambiguous association handling module, and a track management module.

A. Clustering

The clustering module takes the occupancy/velocity grid of the BOF as the input and extracts object-level reports from it. Its main ideas are:

- 1) using the prediction result of the tracking module to define a region of interest (ROI) allowing to avoid searching in the complete grid,
- 2) using both occupancy and velocity estimates in order to better separate/associated the extracted clusters.

A natural algorithm to extract a cluster in a grid is to start from a given cell (pixel) and expand the cluster by deciding, for each eight-neighbor cell, whether the neighbour is to be included or not according to a connectivity criterion.

In order to avoid searching for clusters in the whole grid, we use the predicted targets' states as a form of feedback. For a given target T_i , the predicted state is used to define a region of interest ROI_{T_i} in the BOF grid. ROI_{T_i} is used as the search region in which the clustering module will try to extract a cluster (report) to be associated implicitly to T_i .

To take advantage of occupancy and velocity estimates provided by the BOF grid, the used connectivity criterion is as follows. Starting for a given cell c for which $P(O_c^t) \geq occ_threshold$, a neighbouring cell n is added to the cluster if and only if:

- $$\begin{cases} - \text{Cell } n \text{ is not associated yet (included in no cluster),} \\ - P(O_n^t) > occ_threshold, \\ - MahDist(P(V_c^t), P(V_n^t)) < vel_threshold, \end{cases}$$

where $MahDist(P(V_c^t), P(V_n^t))$ is the Mahalanobis distance between the velocity distributions of a two neighbour cells.

If the "Cell n is not associated yet" predicate is not respected, the corresponding cell is tagged as an ambiguous-associated one. This corresponds to an ambiguous association case which need to be dealt with in a special manner (III-B). If such a situation is not encountered, the extracted cluster is implicitly associated to the target T_i defining the used ROI_{T_i} .

Simple statistics are then performed on the extracted cluster in order to obtain a 4-dimensional observation corresponding to cluster's position and velocity. Both the position and velocity components are represented as 2D Gaussian distributions (mean vector and covariance matrix).

B. Re-clustering and tracks merging

During the clustering process, three possible situations need to be considered (Fig. 6).

- **Case 1:** no cell with $P(O_c^t) > occ_threshold$ is found in ROI_{T_i} defined by the considered target T_i . The target T_i has not been observed and no association is needed.
- **Case 2:** a cluster C is extracted. It's implicitly associated to the target T_i defining the used ROI_{T_i} . This situation occurs when there is no ambiguity in the association. This is an advantageous situation allowing a fast clustering-association procedure. Fortunately, this

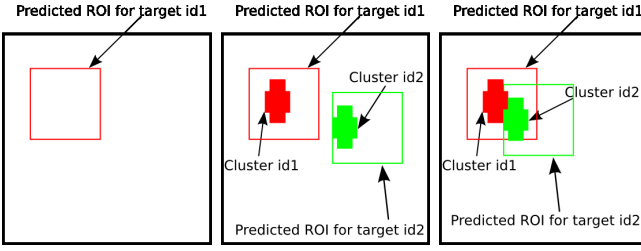


Fig. 6. Situations occurring during the clustering/association process.

case is the most frequent one when using the algorithm to the real data sets.

- **Case 3:** a set of cells c_i having $P(O_{c_i}^t) > occ_threshold$ and $MahDist(\cdot) < vel_threshold$ are extracted in ROI_{T_i} . However, they have already been assigned to other targets. In this conflicted case, an observation (cluster) could be possibly generated by two (or more) different targets.

The first two cases are normal cases, however, the third case is referred as an ambiguous association case which need to be dealt with in a special manner. The ambiguous association could occur in the following two situations:

- Different targets are being too close to each other and the observed cluster is in fact the union of the more than one observations generated by different targets.
- The different tracked targets are corresponding to a single object and should be merged into one.

We take a re-clustering strategy to deal with the first situation and a cluster merging strategy to deal with the second one.

When an ambiguous association occurs, a set of tracks T_1, T_2, \dots, T_m are identified as the potential candidates to be associated to the extracted cluster. We have to cut up this cluster and generate a sub-cluster (possibly empty) for each candidate. This re-clustering is achieved by applying a simple Mahalanobis distance between the considered cell and the centre (mean position) of each target $T_k, k = 1 \dots m$.

To deal with the second cause of the ambiguous association, we introduce a concept of “alias” which is in the form of a two-tuples to represent the duplicated tracks. When an ambiguous association between two tracks T_i and T_j is detected, an alias $ALIAS(T_i, T_j)$ is initialized and added to the potential aliases list. At each time frame, the tracker updates this list by confirming or disproving the existence of each alias hypothesis $ALIAS(T_i, T_j)$ according to the observation of the ambiguous association. If an ambiguous association occurs between T_i and T_j , and the alias $ALIAS(T_i, T_j)$ is found in the potential alias list, the probability $P^t(S(T_i, T_j))$ is updated by a confirming step using a Bayesian filtering approach as follows:

$$P^t(S | F) = \frac{P^{t-1}(S) \times P(F | S)}{P^{t-1}(S) \times P(F | S) + [1 - P^{t-1}(S)] \times P(F | \neg S)},$$

where:

- $S \equiv$ “the T_i and T_j tracks are alias for the same object”.
- $F \equiv$ “an ambiguous association between the tracks T_i and T_j is observed”.

The probability values $P(F | S)$ and $P(F | \neg S)$ are constant parameters of the tracker. The former denotes the probability of observing an ambiguous association when the two concerned tracks are alias of the same object and is set to a constant value 0.8. The second denotes the probability of falsely observing an ambiguous association and is set to 0.1.

When $ALIAS(T_i, T_j)$ is found in the potential alias list but is not observed as an ambiguous association, its probability is disproved in a similar manner:

$$P^t(S | \neg F) = \frac{P^{t-1}(S) \times P(\neg F | S)}{P^{t-1}(S) \times P(\neg F | S) + [1 - P^{t-1}(S)] \times P(\neg F | \neg S)}.$$

Then, according to the probability $P^t(S(T_i, T_j))$, the decision of merging of tracks T_i and T_j could be considered. The merging decision is done by comparing the actual Mahalanobis distance between T_i and T_j to a given threshold.

C. New tracks creation

For new targets creation, we introduce a concept “cluster seed” to define a cell in the BOF grid where we will try to find, for each step, a new (non-associated) cluster. Indeed, the searching for potential new targets is performed after all the existing tracks are processed. Thus, only non-associated cells will be processed to extract clusters as the observations for the potential new targets. The “cluster seed” concept is general and can be implemented via various strategies. The simplest strategy is to insert a possible seed in each cell of the grid. However, more sophisticated strategies could be more efficient. For example, cluster seeds could be inserted only in entrance regions of the monitored area.

D. Tracks updating and deleting

The prediction and estimation of the targets are accomplished by attaching a Kalman filter [8] with each track. Once associated to a given track, a report (Gaussian distributions for both position and velocity) corresponding to an extracted cluster is used as an observation to re-estimate the position and velocity of the track in a prediction-update step. For non-observed tracks, only a prediction step is taken by applying the dynamic model to the estimation result of the precedent time step.

The deleting of tracks is also achieved in a Bayesian manner. If an existing track T is associated with a given report (cluster), its existence probability is increased using the following formula:

$$P^t(E | O) = \frac{P^{t-1}(E) \times P(O | E)}{P^{t-1}(E) \times P(O | E) + [1 - P^{t-1}(E)] \times P(O | \neg E)},$$

where:

- $E \equiv$ “the target T exists”.
- $O \equiv$ “the target T has been observed (associated)”.

The parameters $P(-O | E)$ and $P(O | -E)$ are the tracker miss-detections and false alarms probabilities respectively.

If an existing target is not associated with any report (cluster), its existence probability is decreased in the similar way:

$$P^t(E | -O) = \frac{P^{t-1}(E) \times P(-O | E)}{P^{t-1}(E) \times P(-O | E) + [1 - P^{t-1}(E)] \times P(-O | -E)}.$$

According to the existence probability, the track deleting operation is achieved by applying a deleting threshold on it.

IV. EXPERIMENTAL RESULT

The proposed approach has been applied in several driving assistance projects and achieved satisfied results in conditions including both highway and cluttered urban environments. The used sensor modalities include:

- multi-layer lidars,
- Computer Vision detection algorithms (Fig. 1),
- Stereovision-based 3D sensors.

However, according to the confidentiality agreements of the on-going projects, the results could not be published. Here, we provide some recent experiment results on our Cycab platform.

The Cycab platform is equipped with SICK lidar, GPS, mono-camera and odometer. To demonstrate the accuracy of the tracking algorithm, we used several GPS which are carried by pedestrians or vehicles. Because the experiments were carried out in the parking area of Inria Rhone-Alpes, within this limited range, the precision of the GPS is highly reliable, which provides us the ground truth of the locations of the moving objects. During the experiment, the SICK lidar served as the main sensor. The camera data was not used by the algorithm right now.

The object of the proposed algorithm is to track the moving objects in scene robustly and efficiently. However, in a normal environment (except the extremely cluttered environment) most of the sensor readings come from static objects. Thus, if we update the BOF with all the lidar data and apply the tracking algorithm directly, large amount of static objects will be detected and tracked. Basically we could apply two different straightforward methods to overcome this problem. The first idea is to remove objects with a speed below a given threshold. This could be achieved by making use of the velocity estimates of the objects given by the tracker and the ego-motion estimate provided by the odometer model. Unfortunately, the combination of the estimated ego-velocity and the targets velocities is not accurate enough, which leads to removing the low speed moving objects, i.e. pedestrians, by mistake.

The second idea is to divide the lidar data into a static set and a dynamic set, and only use the dynamic set to update the BOF. We applied the second method in the experiment. The division of the lidar data is achieved by maintaining a



Fig. 7. Experiment scene.

well discretized occupancy grid map [5] centred at the Cycab and moved along with it. Each cell in this map represents an occupancy probability. If the occupancy probability exceeds a given threshold, this cell is regarded as a static cell, and the lidar data fall into this cell are removed from updating of the BOF. Different from the BOF, the occupancy grid map is implemented in the global coordinates. Thus, the ego-motion of the Cycab is also needed to be estimated. We applied the odometer motion model to predict the location and used an iterative closest point (ICP) [9] algorithm to update it. In our experiment, this scheme has shown high accuracy.

We first apply our algorithm to detect and track a car which moves in front of the Cycab in the same direction as shown in Fig. 7. The result of dividing of the lidar data into dynamic data set and static data set is shown in Fig. 8. The first row of the figures corresponds to the data set of the SICK lidar used to update the BOF. The second row of the figures corresponds to the visualization of the Bayesian occupancy filter. The colour of the cells represents the occupancy probability. The third row gives out the tracker output from the fast clustering-tracking algorithm. The scale of the ellipse represents the uncertainty of the tracked target position. The small arrow start from the centre of the eclipse gives the estimated velocity of the target relative to the Cycab. The first column in Fig. 8 shows the results using the full dataset as the input to the BOF and tracker. There exist about 10 targets being detected and tracked. However, only one of them is the real moving object we are interested in. The second column shows the results using only the dynamic dataset obtained from the aforementioned data division algorithm. It is clearly shown that all the static objects are removed, and the moving object is correctly tracked.

The result shown in Fig. 9 to Fig. 11 demonstrates the accuracy of the tracker compared with a NNJPDA tracker. The moving car before the Cycab is detected and tracked consistently for 35 seconds. The relative position between the Cycab and the target comes from the GPS data is taken as the ground truth and is compared with that estimated by the trackers. The comparison of x relative position is shown in Fig. 9, while the y relative positions are compared in Fig. 10. The distance error of the estimated target position to

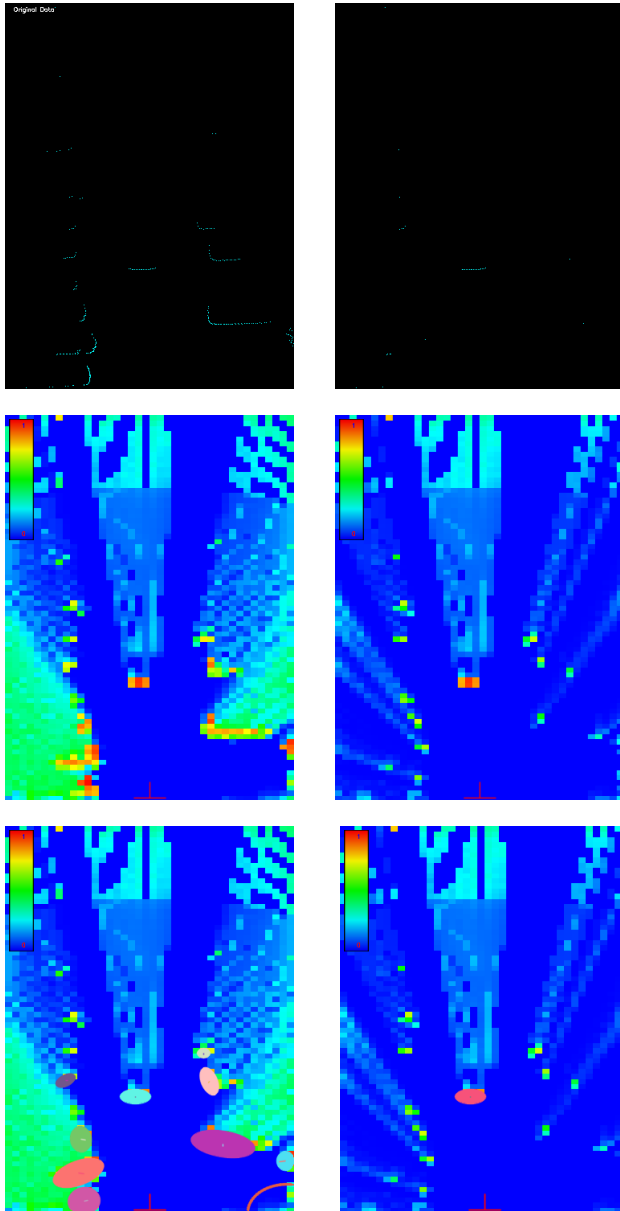


Fig. 8. Comparison of the results with and without dividing the lidar data into static and dynamic data sets.

the GPS data is shown in Fig. 11. As could be seen in the figures, our algorithm succeeded in tracking of the target consistently. However, from 21 seconds to 22 seconds, the NNJPDA tracker lost the target. The average distance error is 0.39 meters for our algorithm compares with 0.37 meters of the NNJPDA tracker. Consider the scale of the target car (roughly 1.5 meters by 2.5 meters), these results show that the precision of both the algorithms are satisfied in this application.

The implementation of the BOF and the tracker is in the C++ programming language without optimization. Experiments were performed on a laptop with an Intel Centrino processor with a clock speed of 1.6GHz. The time consumption of the grid map methods depends on the discretization

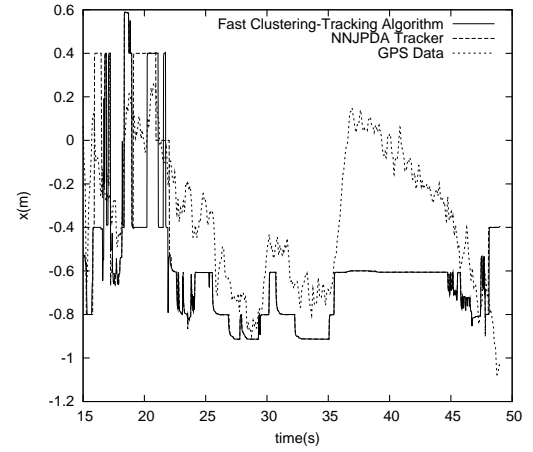


Fig. 9. The x relative position of the target compared with the GPS data.

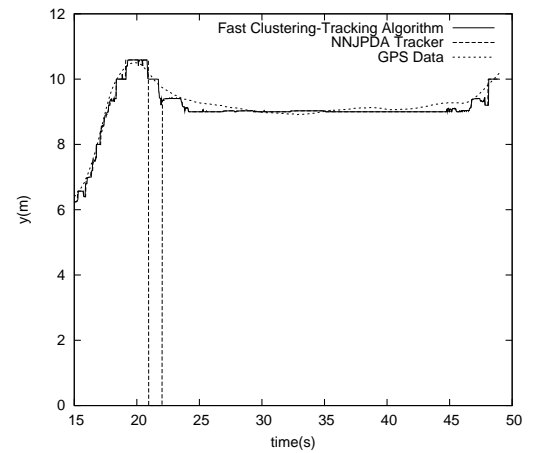


Fig. 10. The y relative position of the target compared with the GPS data.

and the discretization of the velocities. In our experiment, we represent the ground plane with a dimension of 30 meters by 16 meters, with a discretization resolution 0.4 meters by 0.4 meters. The occupancy grid map represents the ground plane around the vehicle with a dimension of 30 meters by 30 meters, with a discretization resolution 0.15 meters by 0.15 meters. The algorithm processes with an average frame rate of 6.2 frames/sec. The BOF consumes with an average of 0.11 seconds per frame, while the ICP algorithm uses up to 0.017 seconds and the updating of the occupancy grid map uses up to 0.078 seconds per frame. The average time consumption of the fast clustering-tracking algorithm is roughly 0.0003 seconds per frame and increases linearly with the number of targets in scene which could be discarded compared with that of the NNJPDA tracker. The time efficiency of the NNJPDA tracker is highly depended on the number of the targets being tracked and the clusters extracted from the output of the BOF. When there exist an average of 11 targets and 18 clusters, the NNJPDA consumes 0.075 seconds per frame. However, this number increases drastically to 5 seconds per frame, when the number of the targets and the clusters increase up

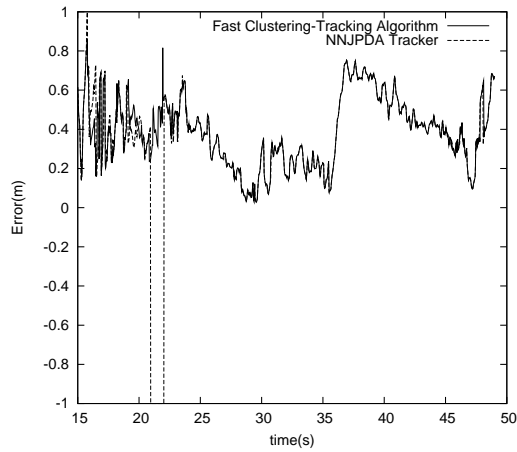


Fig. 11. The distance error of the target to the GPS data.

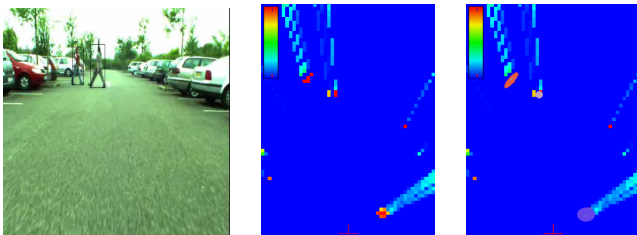


Fig. 12. The two persons are walking towards each other in the perpendicular direction to the Cycab.

to 22 and 28 accordingly. This phenomenon is caused by the combination explosion in the algorithm which produces sparingly hypotheses that need to be processed. The experimental results show that both our algorithm and the classical NNJPDA tracker managed to track the targets accurately and consistently. However, compared with the classical NNJPDA tracker, the fast clustering-tracking algorithm is far more efficient so as to be suitable for cluttered environment.

Another experiment is shown from Fig. 12 till Fig. 14 in time sequence. Two pedestrians walked in the direction perpendicular to the moving direction of the Cycab. The first columns of the figures are the corresponding camera images in which the target is shown by the bounding box schematically (because of the limitation of the camera's field of view, there exists a third pedestrian which can not be seen in the image). The second columns show the outputs of the Bayesian occupancy filter. The third columns are the tracking results. The uncertainty of the tracked target is also fitted into a Gaussian distribution and is represented by an ellipse. In Fig. 12 the pedestrians are properly tracked. In Fig. 13, an occlusion occurs, one of the targets begin to disappear because of not being associated with any extracted clusters. In Fig. 14, after the occlusion occurs for several frames and finishes, both of the targets are detected and tracked again. Note that, the ID of the occluded object remains the same before and after the occlusion, which is shown by the same colour of the drawn target. This means the BOF framework and the proposed tracker are able to manage the

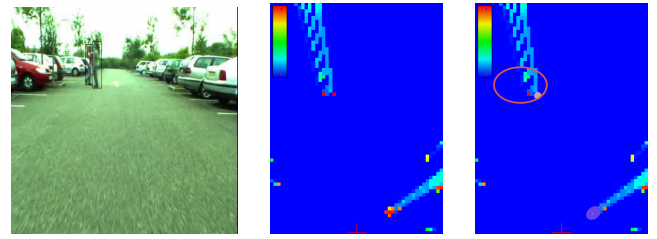


Fig. 13. An occlusion takes place.

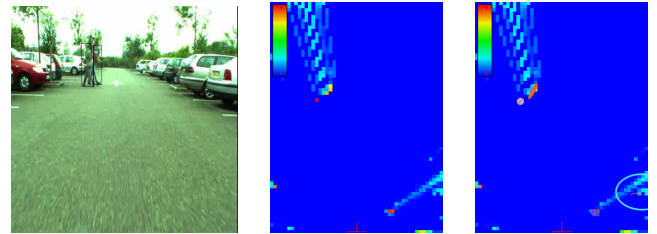


Fig. 14. The occlusion lasts for several frames.

targets properly during the short time occlusion, which is an important characteristic for a wide range of applications.

V. CONCLUSION

In this paper, we presented a novel object tracking algorithm in the BOF framework. This algorithm takes the occupancy/velocity grid of the BOF as input and extracts the objects from the grid with a clustering module which takes the prediction of the tracking module as a feedback to reduce the computational cost. A re-clustering and merging module is proposed to deal with the ambiguous data associations. The extracted objects are then tracked and managed in a probabilistic way. The experiment results show that the presented algorithm is robust as well as computationally efficient. Future work will aim at adding richer sensory data including the IBEO multi-layer laser range finder and computer vision-based detectors to the proposed framework.

REFERENCES

- [1] M.K. Tay, K. Mekhnacha, C. Chen, M. Yguel, C. Laugier. "An Efficient Formulation of the Bayesian Occupation Filter for Target Tracking in Dynamic Environments". *Int. Journal Of Autonomous Vehicles*, 6(1-2):155-171, 2008.
- [2] H.P. Moravec. "Sensor fusion in certainty grids for mobile robots". *AI Magazine*, 9(2), 1988.
- [3] Y.B. Shalom and T.E. Fortman. "Tracking and Data Association". *Academic Press*, 1988.
- [4] C. Coue, C. Pradalier, C. Laugier, Th. Fraichard, and P. Bessiere. "Bayesian occupancy filtering for multitarget tracking: an automotive application". *Int. Journal of Robotics Research*, 25(1):19-30, 2006.
- [5] S. Thrun, W. Burgard, and D. Fox. "Probabilistic robotics", *The MIT Press*, September 2005.
- [6] A.H. Jazwinski. "Stochastic Processes and Filtering Theory". *New York Academic Press*, 1970.
- [7] S.S. Blackman and R. Popoli. "Design and analysis of modern tracking systems". *Norwood, MA: Artech House*, 1999.
- [8] G. Welch and G. Bishop. "An introduction to the Kalman filter". Available at <http://www.cs.unc.edu/welch/kalman/index.html>
- [9] P.J. Besl and N.D. McKay. "A method for registration of 3-D shapes". *IEEE Trans. on Pattern Analysis and Machine Intelligence* 14(2): 239-256, 1992.