Bayesian occupancy filter based "Fast Clustering-Tracking" algorithm

Kamel Mekhnacha, Yong Mao, David Raulo, Christian Laugier

Bayesian occupancy filter based "Fast Clustering-Tracking" algorithm

Kamel Mekhnacha[1], Yong Mao[2], David Raulo[1], Christian Laugier[2]
[1]*Probayes SAS, 38330 Montbonnot Saint Martin, France, E-mail: kamel.mekhnacha@probayes.com*
[2]*INRIA Rhone-Alpes, 38334 Montbonnot, France, E-mail: yong.mao@inrialpes.fr*

***Abstract*— It has been shown that the dynamic environment around the mobile robot can be efficiently and robustly represented by the Bayesian occupancy filter (BOF) [1]. In the BOF framework, the environment is decomposed into a grid-based representation in which both the occupancy and the velocity distributions are estimated for each grid cell. In such a representation, concepts such as objects or tracks do not exist and the estimation is achieved at the cell level. However, the object-level representation is mandatory for applications needing high-level representations of obstacles and their motion. To achieve this, a natural approach is to perform clustering on the BOF output grid in order to extract objects. We present in this paper a novel clustering-tracking algorithm. The main idea is to use the prediction result of the tracking module as a form of feedback to the clustering module, which reduces drastically the complexity of the data association. Compared with the traditional joint probabilistic data association filter (JPDAF) approach, the proposed algorithm demands less computational costs, so as to be suitable for environments with large amount of dynamic objects. The experiment result on the real data shows the effectiveness of the algorithm.**

I. INTRODUCTION

Perceiving of the surrounding physical world reliably is a major demanding of the driving assistant systems and the autonomous mobile robots. The dynamic environment need to be perceived and modeled according to the sensor measurements which could be noisy. This problem is normally treated within the estimation framework. The major requirement for such a system is a robust target tracking system. Most of the existing target tracking algorithms [2] use an object-based representation of the environment. However, these existing techniques have to take into account explicitly data association and occlusion problems which are major challenges of the performances. In view of these problems, a grid based framework, the Bayesian occupancy filter (BOF) [1] [3] [4] has been presented in our previous works.

In the BOF framework, concepts such as objects or tracks do not exist. It decompose the environment into a grid based representation. A Bayesian filter [5] based recursive prediction and estimation paradigm is employed to estimate an occupancy probability and a velocity distribution for each grid cell. Thanks to the grid decomposition, the complicated data association and occlusion problems do not exist. The BOF is extremely convenient for applications where no object-level representation is needed. Another advantage of the BOF is that the multiple sensor fusion task could be easily achieved. In some situations, using information from multiple sensors could provide more reliable and robust information of the environment. However, in traditional multiple sensor fusion techniques, data association problem could be further complicated. The associations between the two consecutive time instances from the same sensor as well as the associations among the tracks of different sensors will have to be take into account at the same time. Fortunately, these difficulties do not exist in the grid based approaches [6] which deal with the data association problems in a more feasible way. Uncertainties of multiple sensors are specified in the sensor models and are fused into the BOF grid naturally with solid mathematical ground.

Fig. 1. Example of BOF output using a computer vision car detector as input (red boxes): The images are provided by a camera mounted on the moving ego-vehicle. The BOF output is projected back on the image. It represents a grid of occupancy probability (blue-to-red mapped color) and the mean velocity (red arrows) estimates.

Fig. 2. The Cycab Platform.

Despite of the aforementioned advantages, a lot of applications demand the explicit object-level representation. In our former work, we suggested a layer structure for these systems. A joint probabilistic data association filter (JPDAF) [2] based object detecting and tracking method was implemented above the BOF layer. However, when there are enormous amount of dynamic objects in the environment, the number of hypothesises generated by the JPDAF increases rapidly, which makes the method suffers from the computational cost. Regarding to this problem, a novel fast object detecting and tracking algorithm is proposed. A simple and naive clustering algorithm is used to extract objects from the BOF grid. By taking the prediction result of the tracking module as a form of feedback to the clustering module, the clustering algorithm avoids searching in the entire grid which guarantees the performance. A re-clustering and merging strategy is employed only when the ambiguous data association occurs. 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).

In classical tracking methodology [2], the problem of data association and state estimation are major problems to be addressed. The two problems are highly coupled with each other and an error in either portion leads to erroneous outputs. The BOF makes it possible to decompose this highly coupled relationship by avoiding the sensor data association problem, in the sense that data association is to be handled at a higher level of abstraction.

We propose to use a hierarchical approach in which two filtering levels are used (Fig. 3):

(i) Robust grid-level sensing.
(ii) Robust object-level tracking.

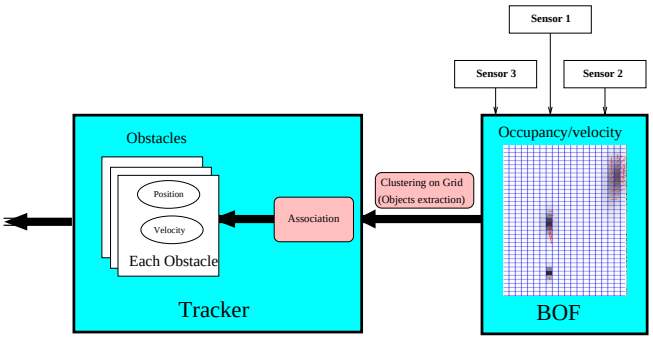

Fig. 3. Sensing/Tracking system architecture.

In a such architecture, the output grid of the BOF grid filter (i.e, the probability distribution on the occupancy of the cell, and the probability distribution on the velocity of the cell occupancy) is used as input for the object tracker by extracting object hypothesis from the grid (Fig. 3).

The paper is organized as follows. In the next section, the BOF framework is briefly described. The object detecting and tracking approach is presented in section III. In section IV, the experimental result of our approach on the real data collected by the Cycab platform is provided. Finally, conclusions are drawn in section V.

II. Bayesian Occupancy Filter (BOF)

The Bayesian Occupancy Filter (BOF) is represented as a two dimensional planar grid based decomposition of the environment. Each cell of the grid contains two probability distributions. A probability distribution on the occupancy of the cell, and the probability distribution on the velocity of the cell occupancy.

Given a set of input sensor readings, the BOF algorithm allows to update the occupancy/velocity estimates of each grid cell.

BOF is a special implementation of the Bayesian filter approach [7]. This approach addresses the general problem of recursively estimating the probability distribution, $P(X^k \mid Z^k)$, of the state X of a system conditioned on its observation Z. This expression is also known as the posterior distribution. The 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 input of the BOF in this case is, for each time step, a set of bounding boxes corresponding to the detected vehicles (red boxes). The output represents a grid of occupancy probability (blue-to-red mapped color) and mean velocity (red arrows) estimates.

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 discredization 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 its all 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 described as follows:

A. Variables

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

- $A_c^t \in \mathscr{A}_c \subset \mathscr{Y}$ represents each possible antecedent of cell c over all the cells in the grid domain $\mathscr{Y}$. The set of antecedent cells of cell c is denoted by $\mathscr{A}_c$ and is defined as a neighbourhood of the cell c.
- $A_c^{t-1} \in \mathscr{A}_c \subset \mathscr{Y}$ is same as A_c^t but for the previous time step.
- $O_c^t \in \mathscr{O} \equiv \{0,1\}$ 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.
- $Z_i^t \in \mathscr{Z}, 1 \leq i \leq S \in \mathbb{N}$, 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:

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

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.
- $P(A_c^t \mid A_c^{t-1})$ 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 assure a perfect *constant velocity hypothesis* between the two time frames $t-1$ and t, this distribution is just:

  $$P(A_c^t \mid A_c^{t-1}) = P(A^{t-1}_{A_c^{t-1}}).$$

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

  Considering imperfect *constant velocity hypothesis* may be done by introducing:
  - $E \in \{0,1\} \equiv$ "There was a prediction error",
  - $P(E) = \varepsilon$ the probability of violating the *constant velocity hypothesis* (a parameter of the model).

  If we define:
  - $P(A_c^t \mid A_c^{t-1}\ \neg E) = P(A^{t-1}_{A_c^{t-1}})$,
  - $P(A_c^t \mid A_c^{t-1}\ E) = \mathscr{U}(A_c^t)$: Uniform predicted antecedent (velocity) when *constant velocity hypothesis* is violated,

  then, $P(A_c^t \mid A_c^{t-1})$ may be written as a mixture:

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

  Which leads to:

  $$\begin{aligned} P(A_c^t \mid A_c^{t-1}) &= (1-\varepsilon) P(A^{t-1}_{A_c^{t-1}}) + \varepsilon\ \mathscr{U}(A_c^t) \\ &= (1-\varepsilon) P(A^{t-1}_{A_c^{t-1}}) + \varepsilon / \|\mathscr{A}_c\|. \end{aligned}$$

- $P(O_c^t \mid A_c^{t-1})$ is the distribution over occupancy given the antecedent of cell c at $t-1$. It represents the prediction (dynamic) model over occupancy. If we assure a perfect *constant velocity hypothesis* between the two time frames $t-1$ and t, this distribution is just:

  $$P(O_c^t \mid A_c^{t-1}) = P(O^{t-1}_{A_c^{t-1}}).$$

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

  When considering imperfect *constant velocity hypothesis*, $P(O_c^t \mid A_c^{t-1})$ may be written as a mixture:

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

  Which leads to:

  $$\begin{aligned} P(O_c^t \mid A_c^{t-1}) &= (1-\varepsilon) P(O^{t-1}_{A_c^{t-1}}) + \varepsilon\ \mathscr{U}(O_c^t) \\ &= (1-\varepsilon) P(O^{t-1}_{A_c^{t-1}}) + \varepsilon/2. \end{aligned}$$

- $P(Z_i^t \mid A_c^t O_c^t)$ 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 $P(Z_i^t \mid O_c^t)$. In the same manner, for sensors providing measurements depending exclusively of velocity, this distribution can be written as $P(Z_i^t \mid A_c^t)$.

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. 4). In this context, the prediction step propagates cell occupancy and antecedent (velocity) distributions of each cell in the grid the 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 $\prod_{i=1}^{S} P(Z_i^t \mid A_c^t O_c^t)$ to obtain the a posteriori state

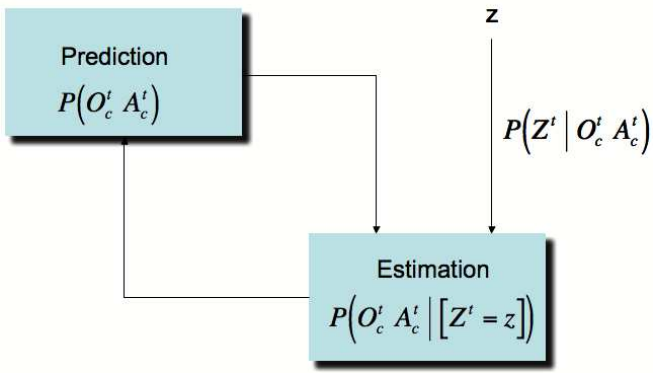

Fig. 4. Bayesian filtering in the estimation of occupancy and velocity distribution in the BOF grids

estimate $P(O_c^t\ A_c^t \mid [Z_1^t \cdots Z_S^t])$. This allows by marginalization to compute $P(O_c^t \mid [Z_1^t \cdots Z_S^t])$ and $P(A_c^t \mid [Z_1^t \cdots Z_S^t])$ that will be used for prediction in the next iteration.

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

$$P(A_c^t \mid [Z_{\text{OCC}}^t]) \propto \sum_{A_c^{t-1} \in \mathscr{A}_c} P(A_c^{t-1}) P(A_c^t \mid A_c^{t-1}) \sum_{O_c^t \in \{0,1\}} P(O_c^t \mid A_c^{t-1}) P([Z_{\text{OCC}}^t] \mid O_c^t). \quad (2)$$

This allows to update the velocity distribution even when no velocity sensors are available. In this case, the update is based exclusively on the occupancy observations.

When an additional velocity sensor $P(Z_{\text{VEL}}^t \mid A_c^t)$ is available, it should be used to update the estimate (2) as follows:

$$P(A_c^t \mid [Z_{\text{OCC}}^t\ Z_{\text{VEL}}^t]) \propto P(A_c^t \mid [Z_{\text{OCC}}^t]) P([Z_{\text{VEL}}^t] \mid A_c^t).$$

III. The "Fast Clustering-Tracking" algorithm

In many applications, the object-level representation are demanded. We propose to use a layer architecture as shown in Fig.3 to obtain this representation. In our former work [1], the object-level presentation is obtained by a classical JPDAF algorithm. However, in the cluttered environment with enormous moving objects, the JPDAF suffers from the combinational explosion of hypothesises. To overcome this problem, we propose a novel object detecting and tracking algorithm. This algorithm could be roughly divided into a clustering module, an ambiguous association handling module and a tracking and 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. A natural algorithm to achieve this is to connect the eight-neighbor cells according to an occupancy threshold *occ_threshold*. In addition to the occupancy values, a threshold of the Mahalanobis distance between the velocity distributions *vel_threshold* is also employed to distinguish the objects that are close to each other but with different moving velocities.

From the implementation point of view, we use an ID grid with the same scale of the input occupancy/velocity grid to store the IDs of the associated targets. The ID grid value is initialized to zero(non-associated) and then is set to the ID of the associated target. 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 with ID *id*, the predicted state is used to define a region of interest (ROI) in which the clustering process starts. After a starting point with an occupancy probability value greater than the *occ_threshold* is found in the ROI, the *id* is propagated in the ID grid using the connectivity criterion among the non-associated cells (cells with $ID = 0$).

A report for the tracker is a 4-dimentional observation corresponding to the position and the velocity of an extracted cluster.

The 2D position component of this vector is computed as the mass center of the region corresponding to the cluster pixels (cells) set. We also compute the corresponding covariance matrix representing the uncertainty of the observed position.

The 2D velocity component is just the weighted mean of the estimated velocities of all cells of the cluster. It comes also with a covariance matrix representing the uncertainty of the observation velocity.

B. Re-clustering and tracks merging

During the clustering process, three possible situation need to be considered.

- **case 1:** no cell with $P(occ = 1) \geq occ\_threshold$ is found. The target has not been observed and no association is needed.
- **case 2:** a cluster C of non-associated cells having $\forall c(i,j) \in C, P(occ(i,j) = 1) \geq occ\_threshold$ is extracted. These cells are then associated to the target *id*: $\forall c(i,j) \in C,\ ID(i,j) = id$, where $ID(i,j)$ is the corresponding ID grid. This situation occurs when there is no ambiguity in the association. This is an advantageous situation allowing a fast clustering-association procedure. Fortunately, this case is the most frequent one when using the algorithm to the real data sets.
- **case 3:** cells having $P(occ(i,j) = 1) \geq occ\_threshold$ exist. However, they have already been assigned to other IDs. 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 refered 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.

Suppose when an ambiguous association occurs, a set of tracks $T_1, T_2, \cdots, T_m$ are the potential candidates to be associated to the the observed cluster. We have to cut up the extracted cluster and generate a sub-cluster (possibly empty) for each candidate. This re-clustering is achieved by a k-means [9] algorithm using a simple Cartesian distance. The considered distance is taken between the center of the sub-cluster and a given cell. In this way, the first cause of the ambiguous association is handled.

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 a potential alias list.

At each 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.

At a given time step t, if the 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:

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

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 $P(F \mid S)$ and $P(F \mid \neg S)$ 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:

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

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 made.

C. 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 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 deleting

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:

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

where:

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

The parameters $P(\neg O \mid E)$ and $P(O \mid \neg E)$ 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:

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

According to the existence probability, the track deleting operation is achieved by applying a deleting threshold on it. However, in order to increase the robustness for occlusions, we also introduce an occlusion calculation procedure. It allows us to decide if a given target is possibly occluded or not. If the target is regarded as being occluded, its existence probability is not decreased even if no observation is associated to it. As a consequence, the target will be kept for longer time without being observed.

E. Tracks updating

In our work, the prediction and estimation of the targets are accomplished by attaching a Kalman filter 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.

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 truths 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 are come from stational objects. Thus, if we update the BOF with all the lidar datas and apply the tracking algorithm directly, large amount of statical objects will be detected and tracked. Basically we could apply two different straight-forward 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 estimations of the objects given by the tracker and the ego-motion estimation given by the odometer model. Unfortunately, because of the BOF has a potentially underestimate the cell velocity characteristic which is caused by the unsynchronized update scheme of velocity layers, and because of the inaccuracy of the ego-motion estimation, this method tends to remove the low speed moving objects, i.e. pedestrians, by mistake.

The second idea is to divide the lidar data into a statical set and a dynamical set, and only use the dynamical set to update the BOF. We applied the second method in the experiment. The division of the lidar data is achieved by maintaining a well discretized occupancy grid map[5] centered at the Cycab and moved along with it. Each cell in this map represents an occupancy probability. If the occupancy probability exceeds an given threshold, this cell is regarded as a statical 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 a global coordinate, 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) [10] algorithm to update it. In our experiment, this scheme has shown high accuracy.

Fig. 5. Experiment scene

We first apply our algorithm to detect and track a car which moves in front of the Cycab with the same direction as shown in Fig. 5. The result of dividing of the lidar data into dynamical data set and statical data set is shown in Fig. 6. The first row of the figures are the data set of the SICK ladar used to update the BOF. The second row of the figures are the visualization of the Bayesian occupancy filter. The color of the cells represents the occupancy probability. The third row give out the tracker output from the fast clustering and tracking algorithm. The scale of the ellipse represents the uncertainty of the tracked target. The small arrow start from the center of the eclipse gives the estimated velocity of the target relative to the Cycab. The first column in Fig.6 show 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 are the real moving object we are interested in. The second column are the results using only the dynamic dataset abtained from the aforementioned data division algorithm. It is clearly shown that all the statical objects are removed, and the moving object is correctly tracked.

The result shown in Fig.7 to Fig.9 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.7, while the y relative positions are compared in Fig.8. The distance error of the estimated target position to the GPS data is shown in Fig.9. 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. Experi-

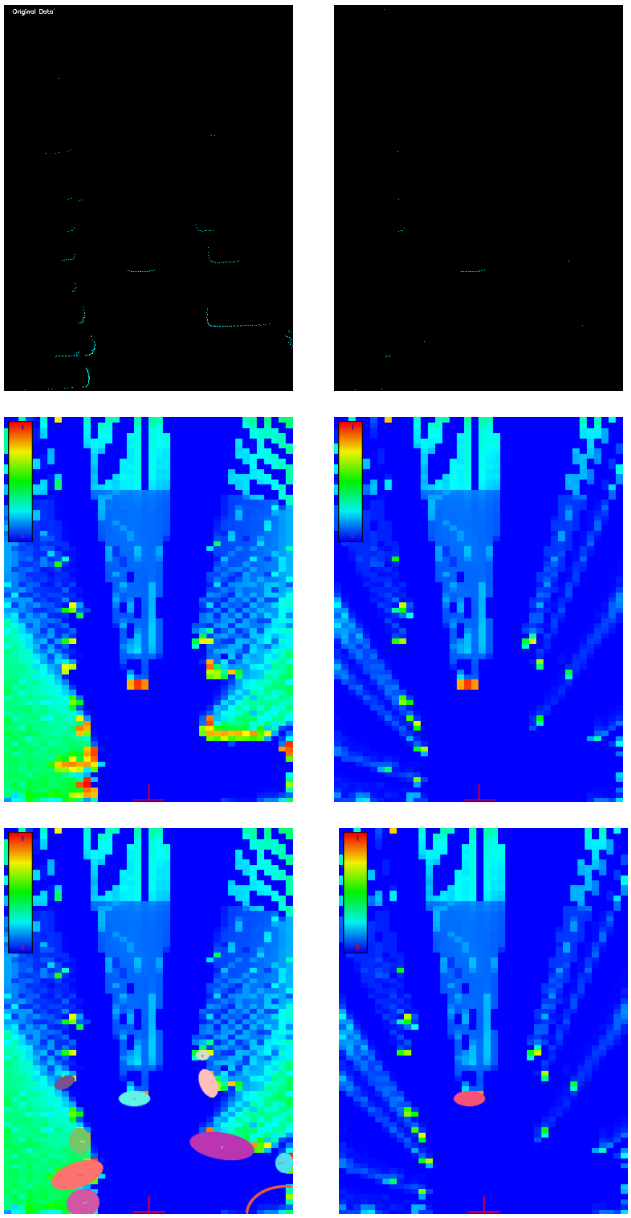

Fig. 6. Comparason of the results with and without dividing the lidar data into statical and dynamical data sets

ments 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 and the discretization of the velocities. In our experiment, we represents 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 0.017 seconds and the updating of the occupancy grid map uses up to 0.078 seconds per frame. The average time consumption

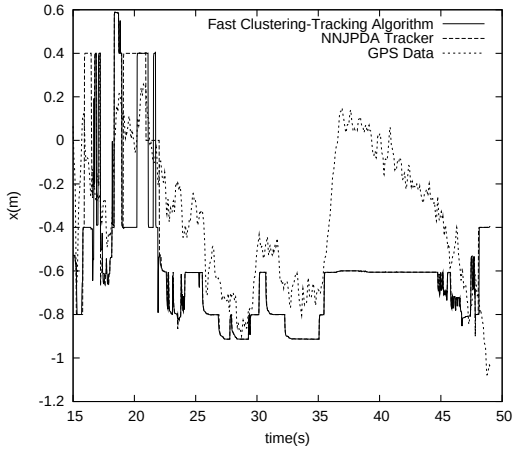

Fig. 7. The x relative position of the target compared with the GPS data

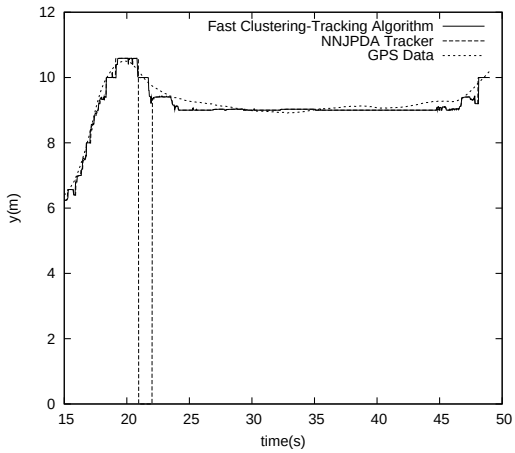

Fig. 8. The y relative position of the target compared with the GPS data

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 to 22 and 28 accordingly. This phenomenon is caused by the combination explosion in the algorithm which produces soaringly hypothesises that need to be processed. The experimental results shows 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. 10 till Fig. 12 in time sequence. Two pedestrians walked in the direction perpendicular to the moving direction of the Cycab's. The

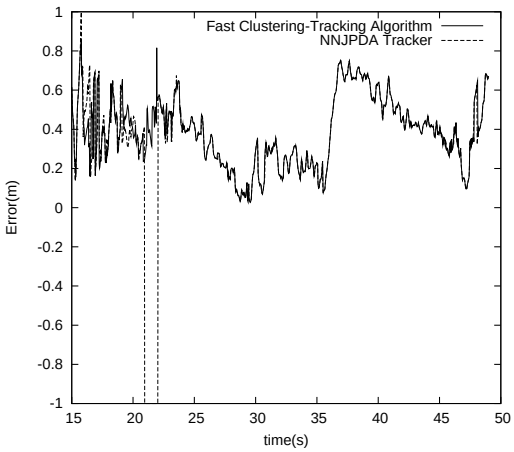

Fig. 9. The distance error of the target to the GPS data.

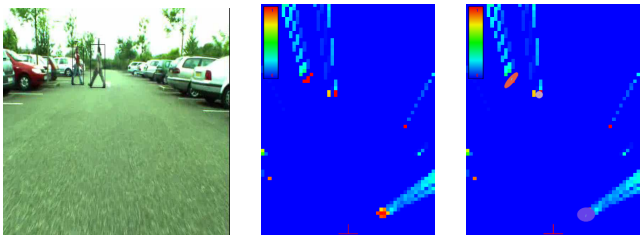

Fig. 10. The two persons are walking towards each other in the perpendicular direction to the Cycab

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 a ellipse. In Fig. 10 the pedestrians are properly tracked. In Fig. 11, an occlusion occurs, one of the targets begin to dispear because of not being associated with any extracted clusters. In Fig. 12, 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 color of the drawn target. This means the BOF framework and the proposed tracker are able to manage the targets properly during the short time occlusion, which is an important characteristic for a wide range of applications.

V. Conclusions

In this paper, we presented a novel object detecting and tracking algorithm for 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

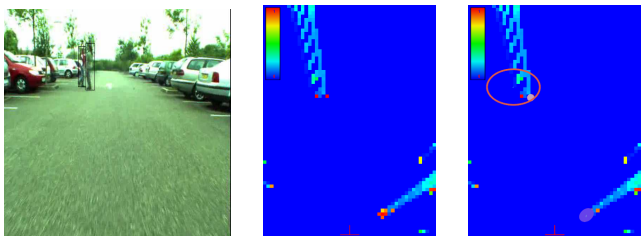

Fig. 11. An occlusion takes place

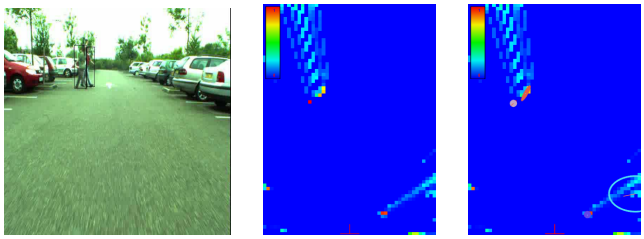

Fig. 12. The occlusion lasts for several frames

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 involves adding richer sensory data including the IBEO multi-layer laser range finder 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". *International Journal Of Autonomous Vehicles* 6(1-2):155-171, 2008.

[2] Y.B. Shalom and T.E. Fortman. "Tracking and Data Association". *Academic Press*, 1988.

[3] C. Coue, Th. Fraichard, P. Bessiere, and E. Mazer. "Multi-sensor data fusion under Bayesian programming: an automative application". *In Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne, (CH), October 2002.

[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, January 2006.

[5] S. Thrun, W. Burgard and D. Fox. "Probabilistic robotics", *The MIT Press*, September 2005.

[6] H.P. Moravec. "Sensor fusion in certainty grids for mobile robots". *AI Magazine*, 9(2), 1988.

[7] A.H. Jazwinsky. "Stochastic Processes and Filtering Theory". *New York Academic Press*, 1970.

[8] G. Welch and G. Bishop. "An introduction to the Kalman filter". Available at *http://www.cs.unc.edu/ welch/kalman/index.html*

[9] C.M. Bishop. "Pattern recognition and machine learning". *Springer*, 2006.

[10] P.J. Besl, N.D. McKay. "A method for registration of 3-D shapes". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(2): 239-256. 1992.