Real-Time Tracking of Moving Objects Using Particle Filters

António Almeida, Jorge Almeida and Rui Araújo
ISR - Institute for Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, P-3030-290 Coimbra, Portugal

Abstract—Mobile robots and vehicles are increasingly used in dynamic environments populated by humans and other moving objects and vehicles. In this context, tracking of surrounding moving objects is important for obstacle avoidance and motion planning. In this paper we present a method for detection and tracking of multiple moving objects using particle filters to estimate the object states, and sample based joint probabilistic data association filters to perform the assignment between the features detected in the input sensor data and filters. Filters management operations are required for appropriate integration of the currently perceived features. A real-time architecture, developed to implement the tracking system, is briefly described. Experimental results obtained with a laser range scanner will be presented demonstrating the feasibility and effectiveness of the presented methods.

I. INTRODUCTION

With the continuous evolution of mobile robotics there will be autonomous vehicles in environments where other moving objects, including humans and, vehicles, evolve. Applications include transportation, home, factory, and office contexts. Learning models of dynamic environments is an important aspect for autonomous robot navigation [1]. Sensor-based methods for modeling and predicting the environment dynamics would increase the degree of anticipatory adaptation to environment evolution. In particular, tracking moving objects is important for motion planning and to anticipate and avoid collisions: giving correct state information of surrounding moving objects to a trajectory planning algorithm would and improve robot motion in dynamic worlds.

One of the problems of tracking is that dynamic objects move in patterns that are highly non-linear. Another important problem is simultaneously tracking multiple moving objects from a common set of sensor data. The object state estimation problem has been tackled using Kalman filters or extended Kalman filters (e.g. [2]). Kalman filters give optimal estimates for linear system and measurement models compounded with unimodal Gaussian noises. However, the Kalman approximation is often not accurate enough to model the non-linear, non-Gaussian, multi-modal characteristics of the system (object(s)) and sensors present in tracking. Extended Kalman filters permit the approximation of non-linear problems by linear models. Recently particle filters were introduced to estimate states for problems with non-linear non-Gaussian process and measurement models [3], [4].

This paper presents a method for tracking in real-time multiple moving objects in dynamic environments using particle filters. Particle filters are based on probabilistic representations of states by a set of samples (particles), with the advantage of making possible the representation of non-linear system and measurement models, and multi-modal non-Gaussian density states. For tracking several moving objects using a common sensor data set, a Sample-based Joint Probabilistic Data Association Filters (SJPDAF) algorithm [5] is used to estimate assignment probabilities between isolated segments on the perceived sensorial data vector (features), and the objects moving on the sensory persectual range.

Section II briefly overviews particle filters, and Section III presents the SJPDAF framework. Section IV presents details about perception and tracking. Section V presents the real-time implementation architecture. Section VI presents experimental results demonstrating the feasibility and effectiveness of the presented methods. Section VII makes concluding comments.

II. ESTIMATION BY PARTICLE FILTERS

Particle filters are state estimation methods for systems with non-linear process and measurement models corrupted with noise which may be non-Gaussian and multimodal. These are recursive Monte Carlo (MC) statistical computing methods. Particle filters are an important alternative to Kalman filters which are optimal to linear systems corrupted with Gaussian noise.

Several methods for position determination and navigation use particle filters [6]. In this paper we describe the application of particle filters for tracking moving objects, where estimation of objects positions are based on measurements from a laser range finder.

The key idea of particle filters is to represent and maintain the posteriori density function by a set of random samples with associated weights and to compute the state estimate from those samples and those weights. As the number of samples becomes very large, this MC characterization becomes an equivalent representation to the usual function description of the posteriori probability density function (PDF), and the method approaches the optimal Bayesian estimate.

Let \( \{ x_{i,k} \}^{N_k}_{i=1} \) denote a random measure that characterizes the posteriori PDF \( p(x_{0:k}|z_{1:k}) \) where \( \{ x_{i,k} \}^{N_k}_{i=1} \) is a set of points with associated weights \( \{ \omega_i^k \}^{N_k}_{i=1} \) and \( x_{n,k} = \{ x_j, j = 0, \ldots, k \} \) and \( z_{1:k} = \{ z_j, j = 1, \ldots, k \} \) are the set of all states and measurements up to time \( k \), respectively.
In our implementation, the importance density is $p_\omega$, where $\omega$ is the weight. This way, particles with large weights are selected [7]. After the resampling step, the weights are taken from an importance density $p_\omega(x_k|z_k)$. Resampling [9] is applied. This step permits the reduction of the weights to overcome this issue. The weights are normalized such that $\sum_{i=1}^N \omega^i_k = 1$. This way the posteriori density at $k$ can be approximated as

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^N \omega^i_k \delta(x_{0:k} - x^i_{0:k}),$$  
(1)

and we get a discrete approximation of the true posteriori probability $p(x_{0:k}|z_{1:k})$, where $\delta(\cdot)$ is the Dirac function.

The weights can be chosen using the principle of importance sampling [7], [8]. This principle states that the weights are taken from an importance density. In our implementation, the importance density is $p(x_k|x_{k-1})$ from which we need to take samples. A sample $x^i_k \sim p(x_k|x_{k-1})$ can be generated by first generating a process noise sampling $v_{k-1} \sim p_\beta(v_{k-1})$ and setting $x^i_k = f_k(x^i_{k-1}, v_{k-1})$, where $p_\beta(\cdot)$ is the PDF of $v_{k-1}$. For this particular choice of importance density the weights are given [7] by

$$\omega^i_k \propto \omega^i_{k-1} p(z_k|x^i_k).$$  
(2)

In our implementation, a resampling step (Systematic Resampling [9]) is applied. This step permits the reduction of the effects of degeneracy, observed in the basic particle filter algorithm. The basic idea is to eliminate particles that have small weights and to concentrate on particles with large weights [7]. After the resampling step, applied at every time index, all the particles take the same weight. This way $\omega^i_{k-1} = 1/N \forall i$, and it follows from equation (2) that

$$\omega^i_k \propto p(z_k|x^i_k).$$  
(3)

The algorithm is presented in Fig. 1 and illustrated in Fig. 2.

**III. Sampled-based Joint Probabilistic Data Association Filters**

One way to track various moving objects with particle filters is to estimate mixture of the compounded state of all objects. However, this method becomes impractical even for a small number of objects since computation grows exponentially in the number of objects. This problem can be overcome by tracking the objects in individually. A data association problem arises in this context: to determine which measurement is caused by which object. In this paper we apply Joint Probabilistic Data Association Filters (JPDAF) [5] for this purpose.

The JPDAF algorithm is an extension of Probabilistic Data Association algorithm, that is able to track various moving objects at the same time and with the same set of measurements. The JPDAF, calculates the probabilities of association from the last set of measurements $z_k$ to the various objects. Each object has its prediction and measurement models - state estimation is performed separately for each object.

Let $x_k = \{x^1_k, \ldots, x^T_k\}$ denote the states of the $T$ moving objects being tracked at instant $k$. Each $x^i_k$ is a random variable in state space of a unique object. Let $z_k = \{z^1_k, \ldots, z^m_k\}$ be a set of measurements at instant $k$, where $z^i_k$ is a feature from that set. $Z_k = \{z_k, \ldots, z_T\}$ is the sequence of measurements observed up to instant $k$.

The key idea for tracking is how to associate the observed features to the individual objects.

In the JPDAF model, a joint association event $\theta$ is a set of pairs $(j, i) \in \{1, \ldots, m\} \times \{1, \ldots, T\}$. Each $\theta$ uniquely determines which feature is assigned to which object. Feature $z^j_k$ is used in model currently undetected objects - no feature found for such objects. Let $\Theta_j\iota$ denote the set of all valid joint associations events which assign feature $j$ to the object $i$. At time $k$, the JPDAF computes the posteriori probability that feature $j$ is caused by object $i$ according to

$$\beta_{ji} = \sum_{\theta \in \Theta_j\iota} P(\theta|z_k).$$  
(4)

Assuming that the estimation problem is Markovian and using probability theory, the probability $P(\theta|z_k)$ of an individual joint association event can be calculated according to

$$P(\theta|z_k) = \int P(\theta|z_k, x_k)p(x_k|z_k, Z_{k-1})dx_k.$$  
(5)

Here the state of the objects must be known to determine associations $\theta$. Conversely, $\theta$ needs to be known in order to determine the objects positions. An incremental approximation is applied to overcome this issue. The key idea is to approximate $p(x_k|z_k, Z_{k-1})$ by the belief $p(x_k|Z_{k-1})$ about the predicted state of the objects, i.e. the prediction computed using all measurements per-
received before time-step $k$. According to this, we obtain

$$P(\theta|Z_k) \approx \int P(\theta|z_k, x_k)p(x_k|Z_{k-1})dx_k$$  \hspace{1cm} (6)

$$= \alpha \int P(\theta|x_k)P(\theta|x_k)p(x_k|Z_{k-1})dx_k$$  \hspace{1cm} (7)

where $\alpha$ is a normalizer factor ensuring that $P(\theta|Z_k)$ sums up to one overall $\theta$. The term $P(\theta|x_k)$ corresponds to the probability of the assignment $\theta$ given the current objects states, and is approximated as a constant. Assuming that each feature is detected independently from the others, we get

$$P(z_k|\theta, x_k) = \gamma^{(m_k-|\theta|)} \prod_{(p,q) \in \theta} P(z_k^p|x_k^p)P(x_k^q|Z_{k-1})dx_k^q,$$  \hspace{1cm} (8)

where $\gamma^{m_k-|\theta|}$ is the probability of all false alarms (features without object in a perception cycle) in $z_k$ given $\theta$. Using (8) in (7), and inserting the result in equation (4) we obtain

$$\beta_{ji} = \sum_{\theta \in \Theta_{ji}} \left[ \alpha \gamma^{(m_k-|\theta|)} \prod_{(p,q) \in \theta} P(z_k^p|x_k^p)P(x_k^q|Z_{k-1}) \right].$$

(9)

Once the assignment probabilities are calculated, the updated estimate of the states is obtained as follows

$$p(x_k^j|Z_k) = \alpha \sum_{j=0}^{m_k} \beta_{ji} p(z_k^j|x_k^j)p(x_k^j|Z_{k-1}).$$  \hspace{1cm} (10)

where $p(z_k^j|x_k^j)$ is the measurement model of the system and $p(x_k^j|Z_{k-1})$ is the previous estimate projected to instant $k$ using the system model.

Since particles are used to describe the density function, we use the SJPDAF proposed in [10], so the method can be applied to a discrete representation. The idea is to represent the density $p(x_k^j|Z_k)$ by a set of $N$ random samples, or particles that constitutes a discrete approximation of a PDF. Here, each particle consists on a pair $(x_k^j, \omega_k^j)$, where $x_k^j$ is the state and $\omega_k^j$ is the importance factor. The prediction step of Bayesian filtering is performed by drawing samples from the state computed in the previous iteration and by updating their state according to the prediction model $p(x_k^j|x_k^j-1, t)$. In the correction step, a measurement $z_k$ is integrated into the samples obtained in the prediction step. With sample-based representation the integration of equation (10) can be done by summing over all samples generated after the prediction step, and we get

$$\beta_{ji} = \sum_{\theta \in \Theta_{ji}} \left[ \alpha \gamma^{(m_k-|\theta|)} \prod_{(p,q) \in \theta} \frac{1}{N} \sum_{n=1}^{N} p(z_k^p|x_k^p, z_k^q, n) \right].$$

(11)

With assignment probabilities computed, the weight of each particle can be calculated by

$$\omega_{ji} = \alpha \sum_{j=0}^{m_k} \beta_{ji} p(z_k^j|x_k^j).$$  \hspace{1cm} (12)

$\alpha$ is a normalizing factor, such that all weights sum to one.

IV. PERCEPTION AND TRACKING

A SICK LMS200 laser range finder was used as source of sensory data. The laser was programmed to transmit 361 measurements per scan, evenly distributed in a 180 degrees angular range (0.5 degrees resolution).

Constant resolution grid representations are used in several processing steps for abstracting the sensor measurements to the filter. Space is divided in square cells with 10 cm side.

A. Grid Models

The occupation grids contains an estimate of the occupation probability of each cell, in the current instant. This probability is estimated as follows for each cell $(x, y)$

$$p_{xy} = \psi \frac{N_{xy}}{M_{xy}},$$  \hspace{1cm} (13)

where $N_{xy}$ represents the number of points in the laser measurement vector that is fall inside the $(x, y)$ cell. $x$ and $y$ are the coordinates of the point of the cell that is closest to the origin. $\psi$ is an adjustment factor and $M_{xy}$ is the maximum number of laser points that cell $(x, y)$ can receive. This maximum number depends on the position of the cell with respect to the laser and is calculated as follows:

$$M_{xy} = \frac{180}{|x| + |y| + 1},$$

(14)

where the value 180 is due the laser angular resolution of 180 sensor points on the half laser aperture of 90 degrees. $M_{xy}$ is used in the computation of other grids discussed below. Using (13) and (14) a probability distribution $P(\text{occupied}_{xy}|Z(k))$ is obtained.

The new occupation grid contains the probability of each cell being currently occupied but not occupied in the previous iteration. This grid is constructed by confronting the actual occupation grid with the grid of the last iteration as follows

$$P(\text{new}_{xy}^k|Z(k), Z(k-1)) = P(\text{occupied}_{xy}|Z(k)) \times (1 - P(\text{occupied}_{xy}|Z(k-1))),$$

(15)

where $P(\text{new}_{xy}^k|Z(k), Z(k-1))$ is the probability that an object has moved into cell $(x, y)$ [10].

The occlusion grid has in each cell the probability of that cell being occulted by a object (moving or not) relative to the laser. This grid is used for direct treatment of occulted objects during tracking. Each point of the laser measurement vector is analyzed and the cells containing segments of the point’s occlusion line are updated on the occlusion grid. For each cell $(x, y)$ a probability $P(\text{occlusion}_{xy}^k|Z(k))$ is obtained.

B. Mobile Objects Segmentation

The final step in the perception involves the segmentation of the measurements vector, and the determination of the most closer segments to laser (minimuns) in order to filter out scenario objects [11]. Then, the moving objects are identified by confronting the minimums grid and the new occupation grid. After this process we have a new grid with probability distribution $P(\text{movingobject}_{xy}^k|Z(k))$, that indicates the cells occupied by moving objects. The next step is to group the close minimums that could
belong to the same object. In this way it is possible, for example, to group only in one segment both legs of a persons.

C. Tracking

With SJPDAF theory, it is possible to track various mobile objects using a particle filter for each one. The SJPDAF algorithm returns the assignment probabilities that permits each feature be associated to one filter.

When a new object steps into the laser perceptual range (PR), a new filter must be initialized and starts to track that object. To initialize the new filter around the new object, it is checked which feature does not have a filter tracking it. This verification is performed by analyzing the distances between the mid points of features and particles of the filters. A new filter is associated to feature

$$J_{\text{new}} = \arg\max_{j \in \text{Features}} \min_{i \in \text{Filters}} d(\hat{m}_j, m_i)$$

where $m_{lf}$ and $m_{rf}$ are the mean of feature $j$ and filter $i$, respectively. Particles of the new filter are spread around $m_{lf_{\text{new}}}$. Each particle is a vector containing position, orientation and velocity components: $(x, y, \theta, \dot{\theta})$. This process is iterated until the number of filters equals the number of features.

When an object steps out of the laser PR, one of the active filters must be closed. For each filter, $i$, an accumulated discounted average $\hat{\Lambda}_k^i$ of the particles weights sum before the normalization step $\Lambda_k^i$ is maintained:

$$\hat{\Lambda}_k^i = (1 - \eta)\hat{\Lambda}_{k-1}^i + \eta \Lambda_k^i,$$

where $\eta$ is a constant that regulates the inertia of the process. For each of the $T - n_k$ filters with the least position of $\hat{\Lambda}_k^i$ there is a variable $\varrho^i$ initialized to Max Cont when the feature is no longer perceived on the sensor data, and $\varrho^i$ is decremented while this situation persists. When $\varrho^i$ attains 0, the filter is deactivated. If the feature reappears in the measurements while the filter is in this transition phase, then the filter is not deactivated, and variable $\varrho^i$ is re-initialized to Max Cont. These variables permit that a filter will not be initialized just because a temporary individual feature, or a filter not be closed just because a feature disappears temporarily.

The overall tracking algorithm can be divided into five parts: (i) Prediction step; (ii) Correction step; (iii) Assignment probabilities calculation using (11); (iv) Particles weights update using (12); (v) Resampling step.

The following prediction model was used on the particle filters is described by equations (18)-(21):

$$x_{k+1}^i = x_k^i + x_k^i \varrho_k^i h \cos(\theta_k^i),$$

$$y_{k+1}^i = y_k^i + y_k^i \varrho_k^i h \sin(\theta_k^i),$$

where $h$ is the sample period. Direction parameter $\theta$ and velocity parameter $\vartheta$, at every time index, are affected by an independent Gaussian noise. $\theta$ and $\vartheta$ equations are

$$\theta_{k+1}^i = \theta_k^i + v_1 n_1,$$

$$\vartheta_{k+1}^i = \vartheta_k^i + v_2 n_2,$$

where $n_1$ and $n_2$ are zero-mean Gaussian random processes with unity variance. Factors $v_1$ and $v_2$ adjust the variance for orientation and velocity, respectively.

In the correction step, the calculation of measurements probabilities given the state vector is performed. In this step $p(z_j(k)|X_k^{i,n})$ is calculated. This is needed to compute the assignment probabilities $\beta_{ij}$. For $j = 0$, we have the probability of the object not being detected:

$$p(z_0(k)|X_k^{i,n}) = p(\text{occluded}_k^{i,n}) = P(\text{occlusion}_{x,y}^k [Z(k)],$$

where $x, n \in \text{cell}(x,y)$, i.e. $p(\text{occluded}_k^{i,n})$ represents the probability of particle $n$ from the filter $i$ be occluded at time $k$, as obtained from occlusion grid (see also Sec. IV-A). $p(z_j(k)|X_k^{i,n})$ for $j = 1, \ldots, k$, is obtained from the grid that represents cells occupied by moving object $j$ (cf. Sec. IV-A):

$$p(z_j(k)|X_k^{i,n}) = p(\text{movobject}_{k}^{i,n}) = P(\text{movobject}_{x,y}^k [Z(k)],$$

where $x, n \in \text{cell}(x,y)$, i.e. $p(\text{movobject}_{k}^{i,n})$ is the probability that particle $n$ of filter $i$ is on a moving object position.

V. REAL-TIME ARCHITECTURE

The communication between the PC running the algorithms for tracking moving objects, and the SICK LMS200 laser range scanner was established through a CAN (Controller Area Network) bus. This bus permits the accommodation of the 500Kbaud sensor data transfer rate of the LMS200.

Since the communication with the laser is performed with a RS422 serial channel, a microcontroller-based interface node was implemented for bridging the laser sensor and PC through the CAN bus (Fig. 3). The microcontroller (µC) interface implemented an algorithm to convert CAN format to serial format and vice versa. A protocol was developed to communicate the laser telegrams (messages) between the µC and the PC through the CAN. For protocol management, a header is annexed to the telegram sent from the PC to the laser. The header and telegram form what we call a CAN packet (Fig. 4). Clearly, this header will not be sent to laser. This header consist in four fields (1 byte each): (1) number of CAN messages into which the packet was subdivided for sending from the PC to the µC (the CAN protocol defines a maximum message length of 8 data bytes) (Fig. 4); (2) operation code, which permits the µC to identify the type of the telegram received, so it can proceed with the appropriate actions in the current operation context. (3) communication velocity of the interface serial port, that permits µC to know when to change velocity of its serial port; (4) telegram size to permit the µC to extract the annexed telegram from the received message, so it can
be sent to the laser. To understand the importance of this protocol notice that the laser can function in continuous mode (measurements sending) or in request mode, making the required µC processing to be completely different. On the other hand, if the sent telegram changes the laser communication velocity, the µC has to reconfigure its own serial port speed, for communication to be continued.

In the laser to PC way, the µC only divides the received telegrams from laser in CAN messages (up to 8 bytes) and puts it on the CAN bus for PC reassembly. The overall effect of this real-time communication interface is to permit a totally transparent communication between the PC and the laser, i.e. the communication is performed as if the laser was directly connected to the PC.

VI. EXPERIMENTAL RESULTS

All experiences were performed on an indoor environments where the scenery could be changed from test to test. The laser was placed at a height of 80 cm. The tracking algorithm was implemented with a frequency of 5 iterations per second. The used PC has Pentium III processor at 1 GHz and 512 Mbytes of RAM. The linux operating (kernel 2.24.25) and the Real-Time Application Interface (RT AI 3.0) were used.

A. Tracking Multiple Objects

In this test several people are tracked in order to demonstrate the tracking algorithm.

The sequence of Fig. 5 starts with a single person being tracked. At the third displayed time sample (row) a new object enters the laser PR, a corresponding feature was detected, and a new filter has been started and is already tracking that person. However its particles do not appear in the image according to the count variable explained in Sec. IV. In the last row the new filter is already represented. From row four to row five we can see that the filters interchange their features. This is a result of SJPDAF utilization and by random filters evolution. Fig. 6 presents a situation, observed later on the same experiment, where there are three people in the PR. It can be observed in this row that the particles of the three active filters have lower diversity. One fact that could justify this occurrence is the used Systematic Resampling that lowers the particles diversity and can provoke the degeneration of the filters. When the tracked mobile object has abrupt motion and a few number of particles are predicted inside the feature, the resampling step makes many equal copies. Although the number of filter particles is always the same, low samples diversity is due to particles being superimposed and have the same direction and velocity \((x, y, \theta, \omega)\). Anyway, in subsequent iterations, the randomness introduced in the filter prediction phase, mitigates this problem, and samples diversity is restored. However, there are rare situations in which the particles of a filter have irrelevant weights and the filter gets a high degenerate level. In such situations the filter has difficulties to track the objects and it is necessary to re-initialize the filter.

A second fact that affects the operation of the filters is that the tracked mobile objects are not always detected, as in the case that a person inverts his/her trajectory, thus staying immobile for some instants. In such situations the new occupation grid does not have a corresponding feature present, and the probability \(p(z_k^n|x_k^*\theta^n)\) if affected, which in turn causes an incorrect evolution of particles. Fig. 7 corresponds to this test and specifies the number of features, active filters, and combinations of valid associations made in assignment probabilities computation referenced in Eq. (11). At the iterations corresponding to time instant of Fig. 6 (inside the \([140, 200]\) time interval) an instability in the number of mobile features can be observed. However, these inverted peaks are quick and
Fig. 7. Number of features detected, number of active filters and number of associations during one experiment of tracking five people.

Fig. 8. Sequence of a occlusion situation.

do not influence the number of active filters due the inertia created by the filters closing algorithm (Sec. IV) and this way each feature always has a filter tracking it. Tracking three people the number of combinations of valid associations is 64. This number grows to 7776 when five people are tracked.

B. Occluded Objects

In order to demonstrate the advantage of handling the occlusion of mobile objects, Fig. 8 presents a sequence where a person moves behind a static object. In this situation, the particles of the filter disperse in a random way consistently with the temporary lack of sensory data and corresponding increase of filter state uncertainty. In the first row the filter tracks the mobile object as usual. In the second and third samples, since the feature is not detected, the particles of the filter are assigned to the occlusion feature, and predicted over the corresponding area. Once the feature reappears in the range measurements, the filter rearranges its particles in order to track the feature again with lower uncertainty, as show in sample four.

VII. CONCLUSIONS

This paper has presented probabilistic methods for tracking multiple moving objects, using sensory data from a laser range scanner. The developed system uses SJPDAF to handle the data association problem, and a particle filter to individually track each object. Particle filters have the advantage that they can be applied to non-linear and non-Gaussian systems. A method was also presented for perception of moving objects, and separate moving objects from all the static objects existing in the environment, based in probability occupancy grids and obstacle segmentation. The paper also presented a real-time architecture that was developed to link all system components and implement the tracking algorithms. The architecture permits a completely transparent communication between the laser sensor and the host computer. Future work includes improving particle filter behavior regarding the tracking of highly abrupt motions.

REFERENCES