REAL-TIME TRACKING USING PARTICLE FILTERS AND SAMPLE-BASED JOINT PROBABILISTIC DATA ASSOCIATION FILTERS

ABSTRACT

With the increasing number of mobile robotic applications, interaction with human populated environments is a big problem for the motion and obstacle avoidance planning. A method for helping the robotic system in motion planning is described in this paper. It is represented an algorithm capable of tracking several moving objects in the surroundings of the mobile robot. This method makes use of Particle Filters, capable of modeling non-linear non-gaussian models of moving objects, and a sample based approach of Joint Probabilistic Data Association Filters, to calculate the assignment probabilities of each feature to each Particle Filter. A real-time architecture developed to implement the tracking system is described in detail. Experimental results obtained with a laser range finder will be presented demonstrating the feasibility and effectiveness of the presented methods.

1. INTRODUCTION

There is an increasing requirement for autonomous robotic vehicles to evolve and operate in environments shared with other moving objects such as humans and other moving vehicles. Transportation, home, factory, and office environments are some example applications. For autonomous navigation it is important that the robot continually builds and updates a model of the dynamic environment where it moves and operates. One such important aspect is the prediction of environment dynamics. In this context an important aspect is the tracking of moving objects which would permit the improvement of the anticipatory capability to avoid collisions. In fact, giving correct state information of surrounding moving objects to a trajectory planning algorithm would improve robot motion in dynamic worlds.

One of the problems of tracking is that dynamic objects move in patterns that are highly nonlinear. 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., (Fos et al., 2002)). 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 (Gordon et al., 1993).

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 Filter (SJPDAF) algorithm (Bar-Shalom and Fortmann, 1988) is used to estimate assignment probabilities between isolated segments on the perceived sensory data vector (features), and the objects moving on the sensory perceptual range.

Section 2 briefly overviews Particle Filters, and Section 3 presents the SJPDAF framework. Sections 4 and 5 present details about perception and tracking. Section 6 presents the real-time implementation architecture. Section 7 presents experimental results demonstrating the feasibility and effectiveness of the presented methods. Section 8 makes concluding comments.

2. PARTICLE FILTERS

Particle Filters are Monte Carlo methods. The Monte Carlo methods are numerical methods based on statistical simulation, where a set of random values are used for the simulation.

The Sequential Importance Sampling (SIS) algorithm forms the basis of various sequential Monte Carlo methods, such as the condensation algorithm (MacCormick and Blake, 1999), bootstrap filtering (Gordon et al., 1993), particle filtering (Carpenter et al., 1999), interacting particle approximations (Crisan et al., 1999), and survival of the fittest (Kanazawa et al., 1996). Particle Filters are used for state estimation on systems with nonlinear process and measurement models subject non-Gaussian disturbances. It is a technique to implement a recursive Bayesian filter using Monte Carlo simulations. The key idea of a particle filter is to approximate a posterior density function by a set of weighted samples, “particles”, in order to compute the estimation of the states based on that set of samples. With the increase of the number of particles, the set becomes an approximation of the Probability Density Function (PDF) of the state, and the SIS filter approaches the optimal Bayesian estimate.

Let the posterior PDF \( p(x_{0:k} | z_{1:k}) \) be represented by \( \{ x_{0:k}, \omega^i_{k} \}_{i=1}^{N_k} \), where \( \{ x_{0:k}, i = 1, \ldots, N_k \} \) is a set of samples with associated weights \( \{ \omega^i_k, i = 1, \ldots, N_k \} \) and \( x_{0:k} = \{ x_{j}, j = 0, \ldots, k \} \) and \( z_{1:k} = \{ z_{j}, j = 1, \ldots, k \} \) are the set of all states and measurements up to time \( k \) respectively. The posterior density at instant \( k \) is then approximated by

\[
p(x_{0:k} | z_{1:k}) = \sum_{i=1}^{N_k} \omega^i_k \delta(x - x^i)
\]
where \( \delta(x) \) is the Dirac delta function, and the weights \( \omega^i_k \) are normalized such that \( \sum_{i=1}^{N} \omega^i_k = 1 \). Thus, (1) is a discrete weighted approximation to the true posterior density function, \( p(x_{0:k} \mid z_{1:k}) \).

The weights are chosen following the principle of importance (Doucet, 1998). This principle is based on the following. Suppose that \( p(x) \propto \pi(x) \) is a probability density from which it is difficult to draw samples but from which \( \pi(x) \) can be evaluated, and \( x^i \sim q(x), i = 1, \ldots, N \), are samples generated from a proposal \( q(\cdot) \) called the importance density. In this way a weighted approximation to the density \( p(x) \) is given by

\[
p(x) = \sum_{i=1}^{N} \omega^i \delta(x - x^i),
\]

where

\[
\omega^i \propto \frac{\pi(x^i)}{q(x^i)}
\]

is the normalized weight of the particle. Therefore, if the samples are drawn from an importance density \( q(x_{0:k} \mid z_{1:k}) \), then the weights in (1) are defined by (3) to be

\[
\omega^i_k \propto \frac{p(x^i_{0:k} \mid z_{1:k})}{q(x^i_{0:k} \mid z_{1:k})}.
\]

In the sequential case, at each iteration, we have an approximation to \( p(x_{0:k-1} \mid z_{0:k-1}) \) and want to approximate \( p(x_{0:k} \mid z_{0:k}) \) with a new set of samples.

If the importance density is properly factorized, it can be proved (Arulampalam et al., 2002) that the weight of the particle is given by

\[
\omega^i_k \propto \omega^i_{k-1} p(z_k \mid x^i_k) p(x^i_k \mid z^i_{k-1})
\]

and the posterior filtered density \( p(x_k \mid z_{1:k}) \) can be approximated as

\[
p(x_k \mid z_{1:k}) \propto \sum_{i=1}^{N} \omega^i_k \delta(x_k - x^i_k)
\]

where the weights are defined in (5). Thus, the SIS algorithm consists of a recursive propagation of the weights and particles as each measurement is received.

The SIS algorithm has a degeneracy problem. One of the solutions proposed to solve this problem consists in making a resampling step when necessary. The basic idea of resampling is to eliminate particles that have small weights and to concentrate on particles that have large weights. This step involves the generation of a new set of particles by resampling \( N \) times from the discrete approximation of \( p(x_k \mid z_{1:k}) \) given by (6). The resulting sample is a independently and identically distributed sample from the discrete density (6). Several algorithms have been proposed for the resampling step (Hol, 2004).

Introducing the following two changes on the SIS algorithm, the SIR (Sequential Importance Resampling) is obtained: (1) make the importance density \( q(x_k \mid x_{k-1}^i, z_{1:k}) = p(x_k \mid x_{k-1}^i) \) and (2) making a resampling step every time index. With this choice of the importance density, the weights of the particles can be updated according to

\[
\omega^i_k \propto \omega^i_{k-1} p(z_k \mid x^i_k),
\]

and with the resampling step at every time index the weights of the particles become normalized \( \omega^i_{k-1} = 1/N \) \( \forall i \), thus

\[
\omega^i_k \propto p(z_k \mid x^i_k).
\]

The SIR algorithm has the advantage of weight update simplicity and easy sampling of the importance density.

### 3. Sampled-Based Joint Probabilistic Data Association Filters

One way to track various moving objects with Particle filters is to estimate the compounded state of all objects. However, this method becomes impracticable even for a small number of objects since computation grows exponentially with the number of objects. This problem can be over taken by tracking 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) (Bar-Shalom and Fortmann, 1988) 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 assignment probabilities from the last set of measurements \( z_k \) to the various objects. Each object has its prediction and measurement models - state estimate 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 \( x^j_k \) 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 \( z^j_k \) is a feature from that set. \( z_k = \{z^1_k, \ldots, z^m_k\} \) 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 \( (j, i) \in \{0, \ldots, m\} \times \{1, \ldots, T\} \). Each \( \theta \) uniquely determines which feature is assigned to which object. Feature \( z^j_k \) is used to model currently undetected objects - no feature found for such objects. Let \( G_j \) denote the set of all valid joint association events which assign feature \( j \) to the object \( i \). At time \( k \), the JPDAF computes the posterior probability that feature \( j \) is caused by object \( i \) according to

\[
\beta_{ji} = \sum_{\theta \in G_j} P(\theta \mid Z_k).
\]

Assuming that the estimation problem is Markovian and using probability theory, it can be proved (Schulz et al., 2003) that the assignment probability is given by

\[
\beta_{ji} = \sum_{\theta \in G_j} \left[ \gamma_{ji}^{(m_k-|\theta|)} \prod_{(p,q) \in \theta} \left( p(z^p_k \mid x^p_k) p(x^p_k \mid Z_{k-1}) \right)^{-1} \right].
\]

where \( \gamma_{ji}^{(m_k-|\theta|)} \) is the probability of all false alarms (features without object in a perception cycle) in \( Z_k \) given \( \theta \) and \( \alpha \) is a normalizer factor. Once the assignment probabilities are calculated, the updated estimate of the states is obtained as follows

\[
p(x_k \mid Z_k) = \sum_{j=1}^{m_k} \beta_{ji} p(z_k \mid x^j_k) p(x^j_k \mid Z_{k-1}),
\]
where \( p(x_0^k | x_{-1}^k) \) is the measurement model of the system and \( p(x_1^k | Z_{k-1}) \) 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 (Schulz et al., 2003), so the method can be applied to a discrete representation. The idea is to represent the density \( p(x_1^k | Z_k) \) by a set of \( N \) random samples, or particles that constitutes a discrete approximation of a PDF. Here, each particle consists on a pair \((x_{1}^{i,n}, \omega_{k}^{i,n})\), where \( x_{1}^{i,n} \) is the state and \( \omega_{k}^{i,n} \) is the importance factor. The prediction step of Bayesian filtering is performed by drawing samples from the set computed in the previous iteration and by updating their state according to the prediction model \( p(x_{1}^k | x_{0}^k, t) \). In the correction step, a measurement \( Z_k \) is integrated into the samples obtained in the prediction step. With the sample-based representation the integration of equation (11) can be done by summing over all samples generated after the prediction step, and we get

\[
\beta_{kl} = \sum_{\theta \in \Theta} \left[ \alpha \gamma_{k-1}^{(m_k-\theta)} \right] \prod_{(p,q) \in \Theta} \frac{1}{N} \sum_{n=1}^{N} p(x_q^k | x_{1}^{i,n}) \right].
\]

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

\[
\omega_{k}^{i,n} = \alpha \sum_{j=0}^{N_{1}} \beta_{jl} p(x_{j}^k | x_{1}^{i,n}).
\]

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

4. PERCEPTION

In our implementation a Laser Range Scanner (Sick LMS200) is used to get the sensor data. This sensor gives us the distance from the laser to an obstacle, and was configured to cover an area of 180 degrees with a resolution of 0.5 degrees at a maximum distance of 8 meters. From the measures taken by the laser, a series of probability occupancy grids is constructed in order to identify the moving objects in space.

4.1 Occupancy Grids

Each cell of the grid represents an area of 10cm x 10cm in the real world, and grids with 160 x 80 cells are used to represent all the space perceived by the laser. The occupancy probability of each cell is calculated as follows:

\[
P_{xy} = \psi \frac{N_{xy}}{M_{xy}},
\]

where \( x \) and \( y \) are the coordinates of the point of the cell that is closest to the origin, \( N_{xy} \) represents the number of laser points perceived inside the area of the cell, \( M_{xy} \) is the maximum number of points that fit in cell \((x, y)\), and \( \psi \) is an adjustable factor used to differentiate cells that have at least one point from cells that do not have any point. Factor \( M_{xy} \) depends on the position of the cell with respect to the laser and is calculated as follows:

\[
M_{xy} = \frac{180}{|x| + |y| + 1}
\]

where the value 180 is due the laser angular resolution of 180 sensor points on the half-laser aperture of 90 degrees.

This occupancy grid is combined at each iteration with it’s predecessor, to obtain a new occupation grid (changed cells, having moving objects). This combination is expressed as:

\[
P(\text{new}_{x,y}^k | Z_k, Z_{k-1}) = P(\text{occ}_{x,y}^k | Z_k) \times (1 - P(\text{occ}_{x,y}^k | Z_{k-1})).
\]

This method provides a certain inertia (filtering) in the new occupancy grid calculation in order to help avoiding errors in the feature extraction.

4.2 Segmentation

In order to improve feature extraction, segmentation of the laser data is performed in order to separate the minimum values from the scenario values. This procedure gives us a way of extracting the points corresponding to some object in the range data. After the segmentation step a set of segments is obtained, which consists of a set of points that can be represented in the same form of occupancy grids.

By compounding the results of the segmentation step and the new occupancy grid, the method can identify and extract the moving features \( p(\text{occ}_{x,y}^k) \). This information is then used as measured features for the SJPDAF and tracking approach.

5. TRACKING

For tracking multiple objects with particles filters, in this work one filter is used for each object, with the SJPDAF framework being used for feature to object assignments.

Figure 1 Grid construction sequence.

5.1 Filter Management

It is important to maintain the correct number of active filters. Next, the method used for this purpose is explained. If the number of features is bigger than the number of active filters it is necessary to search for the features that do not have an associated filter. This search is made comparing the location of the mid point of the feature and the mean of the PDF of the particles filter. This is essential for activating a new filter on each new feature. In the procedure of filter activation, the particles of the new filter are spread in the area of the newly detected feature, and a counter variable \( \xi \) is initialized \( \xi := 1 \). \( \xi \) is incremented every time the feature is detected. In order to filter out erroneously perceived features (e.g. sensor noise), only after some time tracking the feature with \( \xi \) having raised above a certain threshold, \( \text{Min}_\text{Cont} \), the filter really activated. Until that moment, the filter is considered to be in an “embryonic” state. If the feature disappears before \( \xi \) reaches \( \text{Min}_\text{Cont} \) the filter does not enter the definite active state.

For the deactivation of the filters a similar procedure is implemented. At
every iteration a variable $\hat{\Lambda}_k^i$, representing a discounted average of the sum of particle weights before the normalization step, $\Lambda_k^i$, of filter $i$ is updated. Since the sum of particle weights decreases every time a filter is tracking a feature not present in measurements, this value is used to deactivate the filters when there are more filters than features. The method for updating $\hat{\Lambda}_k^i$ depends on the weights on the present iteration, and previous value of $\hat{\Lambda}_k^i$:

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

where $\eta$ is a constant that regulates the inertia of the process. For each of the $T - m$ filters with the least value of $\Lambda_k^i$ there is a variable $\vartheta^i$ initialized to $\max_cont$ when the feature is no longer perceived on the sensor data, and $\vartheta^i$ is decremented while this situation persists. When $\vartheta^i$ 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 $\vartheta^i$ is re-initialized $\max_cont$.

### 5.2 State Prediction

To represent the state of one object it is used a quadruple $(x, y, \theta, \vartheta)$ where $x$ and $y$ represent the relative position to the laser, $\theta$ the heading orientation and $\vartheta$ the velocity of the object. For each object the following kinematic model is used as the system (prediction) model:

$$x_{k+1}^i = x_k^i + v_1 n_1$$
$$y_{k+1}^i = y_k^i + v_2 n_2$$
$$\theta_{k+1}^i = \theta_k^i + v_1 n_1$$
$$\vartheta_{k+1}^i = \vartheta_k^i + v_2 n_2$$

where $h$ is the sampling interval, $n_1$ 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.

### 5.3 Assignment Probabilities

After the prediction step, the feature measurement probabilities given the states, $p(z|x_k^i|x_{k}^{i,n})$, can be calculated. For this purpose the occlusion and mobile features grids are used. For the occlusion feature, $j = 0$, this becomes

$$p(z^0(x)|x_{k}^{i,n}) = p(\omega^0_{x,y}|x_{k}^{i,n}),$$

and for the mobile features, $j = 1, \ldots, m_k$:

$$p(z^j(x)|x_{k}^{i,n}) = p(\omega^j_{x,y}|x_{k}^{i,n}).$$

With this result we are now able to apply expression (12) to calculate the assignment probabilities, where $p(z^j(x)|x_{k}^{i,n}) = p(z^j(x)|x_{k}^{i,n})$. The calculation of the assignment probabilities is made covering all possible features-to-objects combinations for each pair $(x, q)$, thus, we are able to compute $\prod_{j=1}^{m_k} \sum_{i=1}^{N} p(z^j(x)|x_{k}^{i,n})$. Introducing the factor due to the false alarms and summing for all the association events that associate feature $j$ with object $i$, a matrix representing the assignment probabilities is obtained.

### 5.4 Weight Calculus

After the assignment probabilities are calculated the new weights of the particles of the filters are calculated by (13). As we don’t know what feature belongs to what object the weight of each particle is obtained by integrating over all the isolated feature probabilities.

To complete the SIR algorithm a resampling step is performed on all filters. Systematic resampling was used in this work (Hol, 2004).

### 6. REAL-TIME ARCHITECTURE

To implement all this various steps of tracking, a Real-Time architecture must be used. Specifically, the architecture links all system components, and organizes all system activities while meeting the timing constraints of the tracking methods and other system components (e.g. sensors). The results presented in the next section were obtained from an implementation that consists of various modules interconnected by various different communication networks as presented in Figure 2. This figure represents a deployment diagram according to the UML (Unified Modeling Language).

The system is built using a PC, where some modules interact among them, and with some other modules outside the PC. The microcontroller module is able to inter-connect between a CAN bus and the LMS 200 connection that only supports serial communications, RS422/RS232. For testing this tracking approach in a dynamic environment a Nomad 200 robot is used, with the connection between the robot and the PC being a wireless communication based on the Ethernet transmission protocol. This connection to the robot permits the transmission of motion commands obtained by a motion controller in the PC and obtaining data from onboard sensors. The Noman200 module present in the PC represents the robot interface for this controller. Like the LMS200 module that represents the interconnections between Laser and PC.

![Diagram of Real-Time Architecture](image-url)
is at risk. After receiving all the CAN messages of one laser measurement, the LMS200 thread passes the measurement information to the Knvlaser module, to be represented in the GUI, and to the Tracking thread. The later is the most important thread of all the architecture because it implements all the algorithm described in the preceding sections. All the activities due by this thread represent the proposed tracking approach using Particle Filters and SJPDAFs.

Figure 4 represents all the interactions and data fluxes between the threads.

7. EXPERIMENTAL RESULTS

The methods presented in this paper were successfully applied to track in real-time multiple moving objects using real world data obtained with a SICK LMS 200 laser range sensor in a dynamic environment. The results were obtained with a static sensor placed at an height of 80 cm from the ground. Each filter uses 1000 particles, the false alarms probability was set to $\gamma = 0.01$. The current implementation used the Linux-RTAI real-time system, and was configured to integrate laser data using a sampling rate of 5Hz. The laser sensor was connected to the computer through a node of a CAN network. At this rate it was possible to simultaneously track 6 moving objects.
In the experiment represented in Fig. 5, 3 persons were tracked. A problem that can be observed in Fig. 5 is the loss of diversity among the particles of each filter. This problem happens in situations of more abrupt object motions and is due to the resampling step: each filter always has the same number of samples, but there are many equal copies (the resampling is trying to favor the smaller number of particles that were able to track the abrupt motion). In subsequent estimation cycles, the prediction randomness independently introduced in each particle in the prediction step tends to restore samples diversity. In rare situations where the system is not able to track highly abrupt motions, an insufficient number of particles are predicted inside the feature. This affects the robust calculation of the assignment probabilities (eq. 12), subsequently preventing the proper calculation of particle weights (eq. 13). In such situations the filter has difficulties to track the objects. For solving this problem, when these situations are detected, the filter is re-initialized.

7.1 Occlusion Handling
The system is able to appropriately track occlusion situations. Fig. 6 illustrates a situation where two persons moves and one cross behind the other. As can be seen, in this temporary occlusion situation, the samples of the particle filter...
start to spread because the uncertainty in the person’s location increases. This fact is consistent with the lack of sensory information. The SJPDAF assigns the occlusion feature to the filter that represents the person’s location. After the hidden person exits the occlusion area, the feature reappears, and the filter samples grouped again to track the feature as can be seen in the last row. If the feature does not reappear after $\text{Max}_\text{Cont}$ the filter is deactivated (Sec. 5.1). Fig. 6 presents the system handling an occlusion situation where both the occluded and occluding objects are moving.

8. CONCLUSIONS

This paper presented a method for tracking multiple moving objects using particles filters and SJPDAFs. 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 system permits the integration of the advantages of both particle filters and JPDAFs. Particle Filters are able to represent arbitrary densities over the state space of the individual tracked objects. The sample-based approach of the JPDAF is able to efficiently handle the data association problem. The paper also described in detail a real-time architecture that was developed to implement the tracking algorithms in a mobile robot experimental setup. This involves linking all system components while meeting the timing constraints of the estimation methods and other system components. Experimental results were presented demonstrating the feasibility and effectiveness of the presented methods. Future work includes improving particle filter behavior regarding the tracking of highly abrupt motions, and to reduce the computational effort to calculate the assignment probabilities with the SJPDAF framework for bigger number of objects.

REFERENCES


