Tracking Multiple Moving Objects for Mobile Robotics Navigation

Jorge Almeida, António Almeida and Rui Araújo
ISR - Institute for Systems and Robotics
Department of Electrical and Computer Engineering,
University of Coimbra, P-3030-290 Coimbra, Portugal
{jorge.almeida, to.almeida}@gmail.com, rui@isr.uc.pt

Abstract

For mobile vehicle navigation in dynamic environments it is desirable that robots have a model of the dynamic aspects of the world. In this paper we present a method for detection and tracking of multiple moving objects using sensor information. The method uses particle filters to estimate objects states, and sample based joint probabilistic data association filters to perform the assignment of features detected from sensor data to filters. A perception mechanism, based on occupancy grids, is presented to distinguish between mobile features and static objects. Experimental results from a real-time implementation using a laser range sensor will be presented demonstrating the feasibility and effectiveness of the presented methods.[1]

1 Introduction

Environments where mobile robots should operate are subject to dynamic changes: learning models in dynamic worlds is an important aspect for autonomous robots [1]. One of the current and future research problems is the development of sensor-based methods for modeling and predicting the environment dynamics, such that the degree of anticipatory adaptation to environment evolution may be increased. Tracking of moving objects is important for obstacle avoidance and motion planning: giving correct state information of surrounding moving objects to a trajectory planning algorithm, would potentially improve robot motion in dynamic worlds.

Many approaches to track multiple objects are based on the Kalman filter. However, Kalman filters are optimal for unimodal Gaussian distributions of state and measurements modeled through linear process and measurement models. In many cases the Kalman approximation is too crude, and not very accurate, to model the non-linear, non-Gaussian, multi-modal characteristics of the system (object(s)) and the sensors.

This paper presents a method for tracking multiple moving objects in dynamic robot environments. The approach is based on Particle Filters [2] and Sample-based Joint Probabilistic Data Association (SJPDAF) [12]. This method is able to deal with the non-linear and non-Gaussian models because it uses a probability density description of the model instead of a linear and gaussian model. The paper also presents a method for perception of dynamic features from data obtained from a laser range sensor.

The paper is organized as follows. Section 2 introduces particles filters. Section 3 presents the JPDAF and SJPDAF framework. In section 4 is presented a Real-Time Architecture that implements the Tracking algorithm. Section 5 presents a method for perception of dynamic objects using a laser range sensor. Section 6 describes details of the tracking method with emphasis on the issues of filter management, state prediction, and filter assignment probabilities. 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 [11], bootstrap filtering [8], particle filtering [4], interacting particle approximations [6], and survival of the fittest [10]. Particle filters are used for state estimation on systems with non-linear process and measurement models subject non-Gaussian disturbances. It is a technique to implement a

[1]This work was partially supported by FCT, Project POSI/SRI/42043/2001.

0-7803-9402-X/05/$20.00 © 2005 IEEE

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

$$p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) = \sum_{i=1}^{N_s} \omega^i_k \delta(\mathbf{x} - \mathbf{x}^i) \tag{1}$$

where $\delta(\cdot)$ is the Dirac delta function, and the weights ω^i_k are normalized such that $\sum_{i=1}^{N_s} \omega^i_k = 1$. Thus, (1) is a discrete weighted approximation to the true posterior density function, $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$.

The weights are chosen following the *principle of importance* [7]. This principle is based on the following. Suppose that $p(\mathbf{x}) \propto \pi(\mathbf{x})$ is a probability density from which it is difficult to draw samples but from which $\pi(\mathbf{x})$ can be evaluated, and $\mathbf{x}^i \sim q(\mathbf{x}), i = 1, \ldots, N_s$ are samples generated from a proposal $q(\cdot)$ called *importance density*. In this way a weighted approximation to the density $p(\cdot)$ is given by

$$p(\mathbf{x}) = \sum_{i=1}^{N_s} \omega^i \delta(\mathbf{x} - \mathbf{x}^i) \tag{2}$$

where

$$\omega^i \propto \frac{\pi(\mathbf{x}^i)}{q(\mathbf{x}^i)} \tag{3}$$

is the normalized weight of the particle. Therefore, if the samples are drawn from an importance density $q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$, then the weights in (1) are defined by (3) to be

$$\omega^i_k \propto \frac{p(\mathbf{x}^i_{0:k}|\mathbf{z}_{i:k})}{q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})}. \tag{4}$$

In the sequential case, at each iteration, we have an approximation to $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{0:k-1})$ and want to approximate $p(\mathbf{x}_{0:k}|\mathbf{z}_{0:k})$ with a new set of samples.

If the importance density is properly factorized, it can be proved [2] that the weight of the particle is given by

$$\omega^i_k \propto \omega^i_{k-1} \frac{p(\mathbf{z}_k|\mathbf{x}^i_k)p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1})}{q(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k)} \tag{5}$$

and the posterior filtered density $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ can be approximated as

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_s} \omega^i_k \delta(\mathbf{x}_k - \mathbf{x}^i_k) \tag{6}$$

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_s times from the discrete approximation of $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ given by (6). The resulting sample is an independently and identically distributed sample from the discrete density (6). Several algorithms have been proposed for the resampling step [9].

Introducing the following two changes on the SIS algorithm, the SIR (Sequential Importance Resampling) is obtained: (1) make the importance density $q(\mathbf{x}_k|x^i_{k-1}, \mathbf{z}_{1:k}) = p(\mathbf{x}_k|\mathbf{x}^i_{k-1})$, 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

$$\omega^i_k \propto \omega^i_{k-1} p(\mathbf{z}_k|\mathbf{x}^i_k), \tag{7}$$

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

$$\omega^i_k \propto p(\mathbf{z}_k|\mathbf{x}^i_k). \tag{8}$$

Systematic *Resampling* Algorithm

- Initialize CDF: $c_1 = \omega^1_k$
- FOR i=2:N_s
  - Build CDF: $c_i = c_{i-1} + \omega^i_k$
- END FOR
- Start at the beginning of CDF: i=1
- Draw a starting sample : $u_1 \sim \mathrm{U}[0, N_s^{-1}]$
- FOR j=1:N_s
  - Move along CDF: $u_j = u_1 + N_s^{-1}(j-1)$
  - WHILE $u_j > c_i$
  - i=i+1
  - END WHILE
  - Assign Sample $x^{j*}_k = x^i_k$
  - Assign Weight $\omega^j_k = N_s^{-1}$
  - Assign Parent $i^j = i$

- END FOR

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

SIR Algorithm

- FOR i=1:N_s
  - Draw: $\mathbf{x}_k^i \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$
  - Calculate $\omega_k^i = p(\mathbf{z}_k|\mathbf{x}_k^i)$
- END FOR
- Calculate total weight:$t = \sum_{i=1}^{N_s} \omega_k^i$
- FOR i=1:N_s
  - Normalize:$\omega_k^i = t^{-1}\omega_k^i$
- END FOR
- Systematic Resampling

---

3 Joint Probabilistic Data Association

The JPDAF (Joint Probabilistic Data Association Filter) is an extension of the PDA algorithm [3] that is able to track various targets at the same time and with the same set of measures. The goal of such a filter is the calculus of the individual association probabilities of a set of features, present in the measurements, with a set of objects/filters present in the sensor perception range.

3.1 JPDAF

It is assumed that a set of T objects are present in the environment, and $\mathbf{x}_k^i$ is the state vector of object i, where $i = 1, \ldots, T$. Let $\mathbf{Z}_k = \{\mathbf{z}_k^1, \ldots, \mathbf{z}_k^{m_k}\}$ be the set of all m_k perceived features at instant k. In the JPDAF framework an association event θ is a set of pairs $(j, i) \in \{0, \ldots, m_k\} \times \{1, \ldots, T\}$, where each θ determines in a unique way which feature is associated with each object. At every time index the JPDAF processes the posterior probability of associations between each feature j and object i, according to

$$\beta_{ji} = \sum_{\theta \in \Theta_{ji}} P(\theta|\mathbf{Z}_k). \tag{9}$$

Using the Bayes rule and the Markov assumption, the probability of one singular association event, $P(\theta|\mathbf{Z}_k)$, can be calculated as

$$P(\theta|\mathbf{Z}_k) = \int P(\theta|\mathbf{z}_k, \mathbf{x}_k) p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{Z}_{k-1}) d\mathbf{x}_k. \tag{10}$$

Now the states estimates are needed to calculate θ, but also θ is needed for estimating the objects states. We can approximate $p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{Z}_{k-1})$ by $p(\mathbf{x}_k|\mathbf{Z}_{k-1})$ that represents the prediction using the measures obtained prior to instant k. Therefore

$$P(\theta|\mathbf{Z}_k) \approx \int P(\theta|\mathbf{z}_k, \mathbf{x}_k) p(\mathbf{x}_k|\mathbf{Z}_{k-1}) d\mathbf{x}_k \tag{11}$$

$$= \alpha \int P(\mathbf{z}_k|\theta, \mathbf{x}_k) P(\theta|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{Z}_{k-1}) d\mathbf{x}_k, \tag{12}$$

where α is a normalizer factor. The term $P(\mathbf{z}_k|\theta, \mathbf{x}_k)$ represents the probability of one observation given the states of the objects and one association between the features and the objects. Assuming that the features are detected independently, this factor becomes

$$P(\mathbf{z}_k|\theta, \mathbf{x}_k) = \gamma^{m_k - |\theta|} \prod_{(p,q)\in\theta} \int p(\mathbf{z}_k^p|\mathbf{x}_k^q) p(\mathbf{x}_k^q|\mathbf{Z}_{k-1}) d\mathbf{x}_k^q \tag{13}$$

where $\gamma^{m_k - |\theta|}$ is the probability of all false alarms (features without object in a perception cycle) in $\mathbf{z}_k$ given θ. From (9), (12), and (13) the assignment probabilities β_{ji} can written as

$$\beta_{ji} = \sum_{\theta \in \Theta_{ji}} \left[\alpha \gamma^{m_k - |\theta|} \prod_{(p,q)\in\theta} \int p(\mathbf{z}_k^p|\mathbf{x}_k^q) p(\mathbf{x}_k^q|\mathbf{Z}_{k-1}) d\mathbf{x}_k^q \right] \tag{14}$$

Using probabilities β_{ji} and the system model $p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)$ it can be shown that the states estimates become

$$p(\mathbf{x}_k^i|\mathbf{Z}_k) = \alpha \sum_{j=0}^{m_k} \beta_{ji} p(\mathbf{z}_k^j|\mathbf{x}_k^i) p(\mathbf{x}_k^i|\mathbf{Z}_{k-1}), \tag{15}$$

where $p(\mathbf{z}_k^j|\mathbf{x}_k^i)$ is the measurement model, and $p(\mathbf{x}_k^i|\mathbf{Z}_{k-1})$ is the estimate predicted using the system model.

3.2 SJPDAF

The method presented before takes in account continuous probability density functions to represent the states. In this work a set of samples (particles) is used to represent this density function. In this case, the assignment probabilities can be calculated using the following sample based version:

$$\beta_{ji} = \sum_{\theta \in \Theta_{ji}} \left[\alpha \gamma^{m_k - |\theta|} \prod_{(p,q)\in\theta} \frac{1}{N} \sum_{n=1}^{N} p(\mathbf{z}_k^p|\mathbf{x}_k^{q,n}) \right]. \tag{16}$$

From these probabilities, the sample weights can be calculated as follows taking into account all isolated features present in the measurement:

$$\omega_k^{i,n} = \alpha \sum_{j=0}^{m_k} \beta_{ji} p(\mathbf{z}_k^j|\mathbf{x}_k^{i,n}), \tag{17}$$

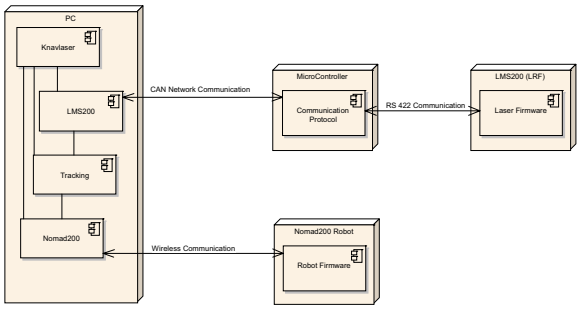


Figure 1. Deployment Diagram.

where α is a normalizer factor in order to make $\sum_{n=1}^{N} \omega_k^{i,n} = 1$. Next, a resampling step is performed on the particle filters and a new iteration is started.

4 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 section 7 were obtained from an implementation that consists of various modules interconnected by different communication networks as presented in Figure 1. This figure represents a deployment diagram according to the UML (Unified Modeling Language) standard. The system is built using a PC, where some modules interact among them, and with some other modules outside the PC. The micro-controller 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.

All the modules inside the PC are connected with shared memories in such a way that the flux of the algorithm could be controlled from the GUI (Graphical User Interface). All the threads enter in Real-Time space after the shared objects have been created. The Knavlaser module represents the GUI and is the one that controls the entire control and data flow of all the system. If the Start button is pressed all the other threads do there work. If the Stop button is pressed all the other threads stop what they are doing and wait for a new command. In the case of Exit all the application is terminated, first returning from Real-Time and then deleting all shared objects.

To communicating with the laser range finder the thread LMS200 starts by configuring the micro-controller in such a way that it reconfigures the LMS200 laser to send continuous measurements of the near space (maximum range: $8\,m$). The micro-controller is also able to temporarily store some of the measures originated from the laser if all the CAN bus bandwidth is at risk. After receiving all the CAN messages of one laser measurement, the LMS200 thread passes the measurement information to the Knavlaser 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.

5 Perception

In our implementation a Laser Range Scanner (Sick LMS200) 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.

5.1 Occupancy Grids

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

$$p_{xy} = \psi \frac{N_{xy}}{M_{xy}}, \tag{18}$$

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 ψ 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:

$$M_{xy} = \frac{180}{|x| + |y| + 1} \tag{19}$$

where the value 180 is due the laser angular resolution of 180 sensor points on the half laser aperture of 90 degrees. Fig. 2 represents and motivates this approach.

This occupancy grid is combined at each iteration with it's predecessor, to obtain a *new occupation grid* (changed

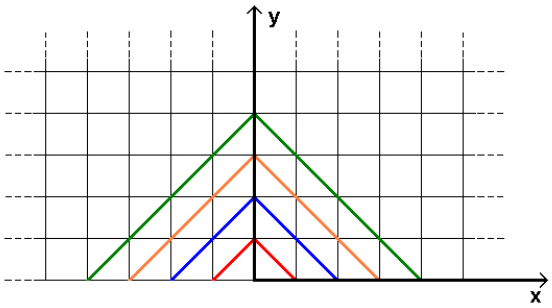


Figure 2. M_{xy} calculation.

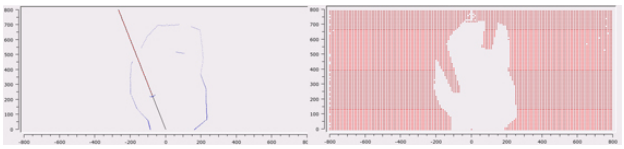


Figure 3. Occlusion grid construction.

cells, having moving objects). This combination is expressed as:

$$P(new^k_{x,y}|\mathbf{Z}_k, \mathbf{Z}_{k-1}) = P(occ_{x,y}|\mathbf{Z}_k) \times(1 - P(occ_{x,y}|\mathbf{Z}_{k-1})). \quad (20)$$

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

An occlusion grid (OG), $p(ocl_{x,y})$, is also required for the tracking system. Each value on the OG represents the probability that a certain cell is occluded and is used for direct processing of occluded objects during the course of tracking. The occlusion map is built using straight lines passing in the laser location and in each sensor point. The probability of each cell beyond the sensor point, that is covered by the corresponding line, is incremented by τ/M_{xy} where τ is an adjusting constant.

5.2 Segmentation

In a 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. For this purpose a method presented in [5] is used. 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 $p(mob_{x,y})$. This information is then used as measured features for the SJPDAF and tracking approach.

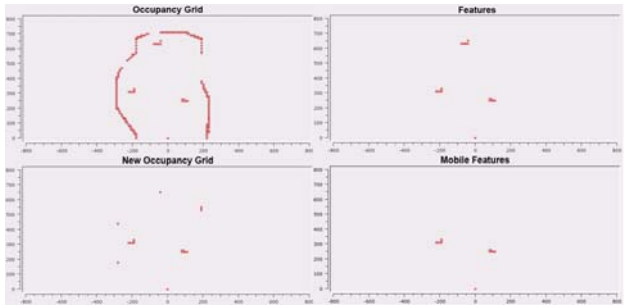


Figure 4. Grid construction sequence.

6 Tracking

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

6.1 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 ξ is initialized $\xi := 1$. ξ 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 ξ having raised above a certain threshold, Min_Cont, is the filter really activated. Until that moment, the filter is considered to be in an "embryonic" state. If the feature disappears before ξ reaches Min_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 $\hat{\Lambda}^i_k$, representing a discounted average of the sum of particle weights before the normalization step, Λ^i_k, 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 $\hat{\Lambda}^i_k$ depends on the weights on the present iteration, and previous value of $\hat{\Lambda}^i_k$:

$$\hat{\Lambda}^i_k = (1 - \eta)\hat{\Lambda}^i_{k-1} + \eta\Lambda^i_k, \quad (21)$$

where η is a constant that regulates the inertia of the process. For each of the $T - m_k$ filters with the least value

of Λ_k^i there is a variable ϱ^i initialized to Max_Cont when the feature is no longer perceived on the sensor data, and ϱ^i is decremented while this situation persists. When ϱ^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 ϱ^i is re-initialized Max_Cont.

6.2 State Prediction

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

$$x_{k+1}^i = x_k^i + x_k^i \vartheta_k^i h \cos(\varphi_k^i) \quad (22)$$

$$y_{k+1}^i = y_k^i + y_k^i \vartheta_k^i h \sin(\varphi_k^i) \quad (23)$$

$$\varphi_{k+1}^i = \varphi_k^i + v_1 n_1 \quad (24)$$

$$\vartheta_{k+1}^i = \vartheta_k^i + v_2 n_2 \quad (25)$$

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.

6.3 Assignment Probabilities

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

$$p(\mathbf{z}^0(k)|\mathbf{x}_k^{i,n}) = p(ocl_{x,y}|\mathbf{x}_k^{i,n}), \quad (26)$$

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

$$p(\mathbf{z}^j(k)|\mathbf{x}_k^{i,n}) = p(mob_{x,y}|\mathbf{x}_k^{i,n}). \quad (27)$$

With this result we are now able to apply expression (16) to calculate the assignment probabilities, where $p(\mathbf{z}_k^p|\mathbf{x}_k^{q,n}) = p(\mathbf{z}^j(k)|\mathbf{x}_k^{i,n})$. The calculus of the assignment probabilities is made covering all possible features-to-objects combinations for each pair (p, q), thus, we are able to compute $\prod_{(p,q)\in\theta} \frac{1}{N} \sum_{n=1}^{N} p(\mathbf{z}_k^p|\mathbf{x}_k^{q,n})$. 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.

6.4 Weight Calculus

After the assignment probabilities are calculated the new weights of the particles of the filters are calculated by (17). 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.

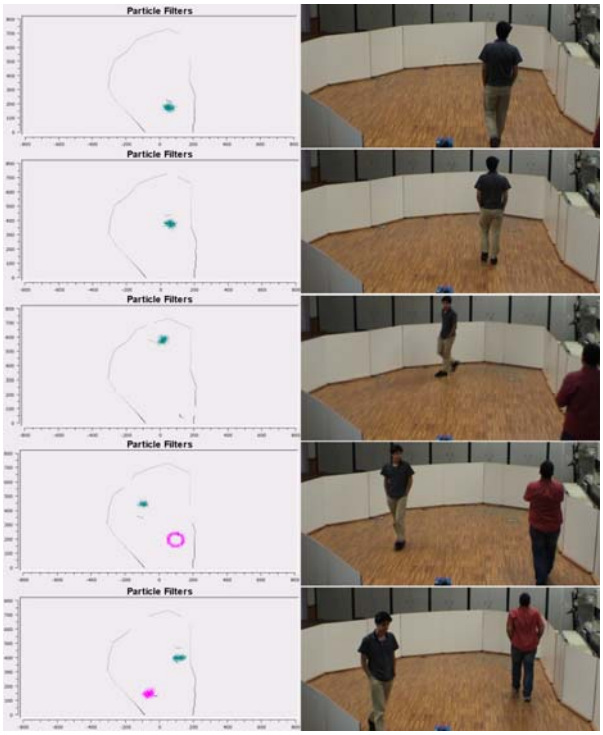


Figure 5. Tracking sequence 1.

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

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 5 Hz. 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.

The experiments are presented as sequences composed of pairs of images: an image that captured the environment, and an image that illustrates the current state of the particle filters at the same instant.

7.1 Tracking Multiple Persons

The first experiment (Fig. 5) starts by tracking only one person, and then another persons enters in the laser perception area. The delay in filter activation (row 3) is due

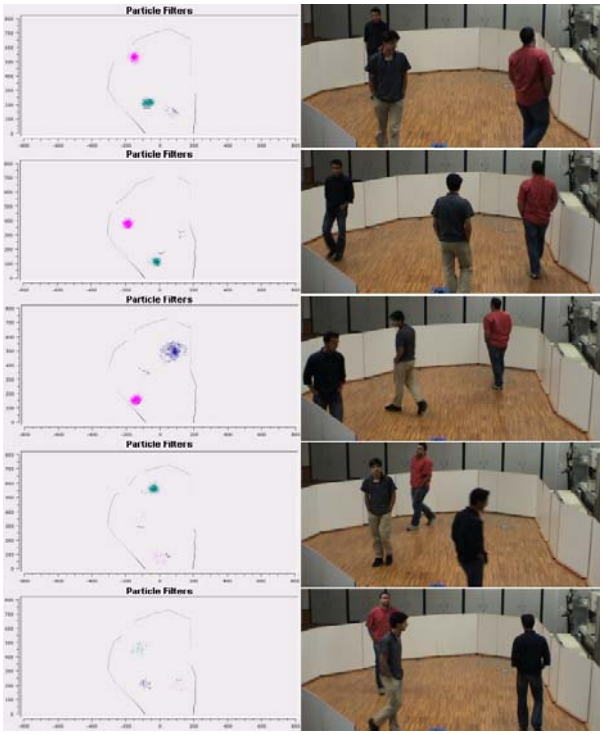


Figure 6. Tracking sequence 2.

to the filter management procedures (Sec. 6.1). In experiment 2 (Fig. 6), 3 persons were tracked. A problem that can observed in Fig. 6 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. (16)), subsequently preventing the proper calculation of particle weights (eq. (17)). 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.

Fig. 7 represents the number of features, the number of filters, and the number of associations during one experiment where the system tracks 5 persons. It can be seen that the number of mobile features is almost always changing, but the number of filters stays constant. This behavior is due to the inertia of the filter deactivation as explained in Sec. 6.1. Fig. 7 also represents the number of association that it's necessary to consider (eq. (16)) when we try to

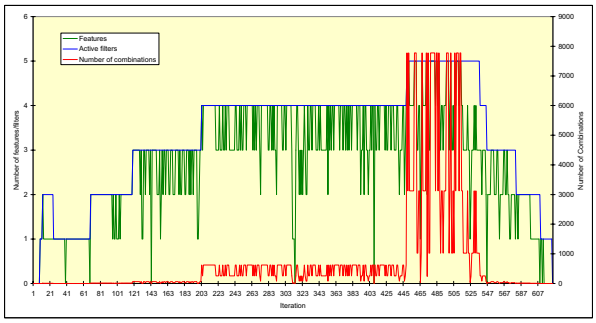


Figure 7. Number of features detected, number of active filters and number of associations during one experiment of tracking five persons.

track five persons.

7.2 Occlusion Handling

The system is able to appropriately track occlusion situations. Fig. 8 illustrates a situation where a person moves through an area that is occluded (behind a static obstacle) from the laser point of view. As can be seen in rows 3 and 4, 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 object exits the occlusion area, the feature reappears, and the filter samples grouped again to track the feature as can be seen in row 5. If the feature does not reappear after *Max_Cont* the filter is deactivated (Sec. 6.1). Fig. 9 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

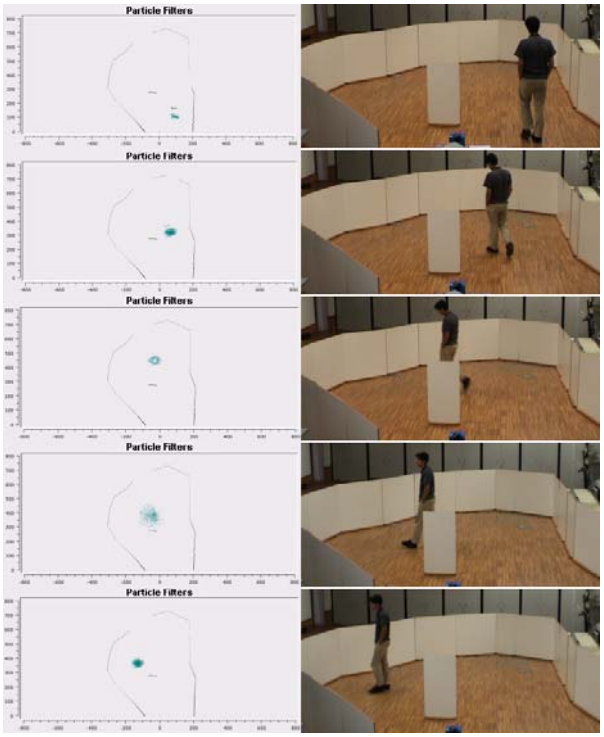


Figure 8. Occlusion sequence 1.

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

[1] R. Araújo, G. Gouveia, and N. Santos. Learning self-organizing maps for navigation in dynamic worlds. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1312–1317, 2003.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE - Transactions on Signal Processing*, 50(2), 02 2002.

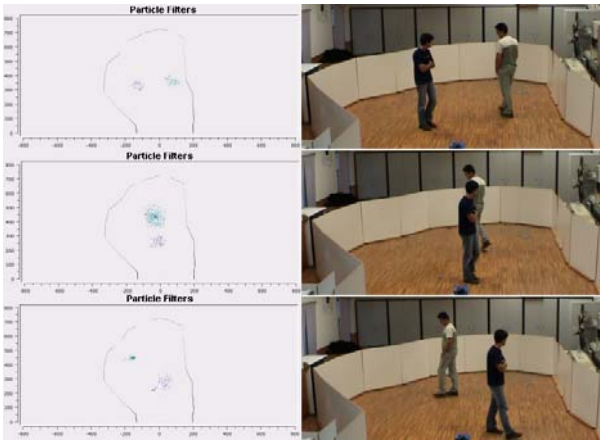


Figure 9. Occlusion sequence 2.

[3] Y. Bar-Shalom and T. E. Fortmann. *Traking and Data Association*, volume 179 of *Mathematics in Science and Engeneering*. Academic Press, 1988.

[4] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. In *Proc. Inst. Electr. Eng., Radar, Sonar, Navig.*, 1999.

[5] D. Castro, U. Nunes, and A. Ruano. Features extraction for moving objects tracking system in indoor enviroments. *IFAC*, 2004.

[6] D. Crisan, P. D. Moral, and T. J. Lyons. *Markov Processes Related Fields*, volume 5, chapter Non-Linear Filtering Using Branching and Interacting Particle Systems, pages 293–319. Publications du Laboratoire de Statistiques et Probabilites, Universite Paul Sabatier, 1999.

[7] A. Doucet. On sequencial monte carlo methods for bayesian filtering. Technical report, Dept. Eng., Univ. Cambridge, UK, 1998.

[8] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. A novel approach to nonlinear/non-gausian state estimation. In *IEE Procedings F*, volume 2, pages 107–113, 1993.

[9] J. D. Hol. Resampling in particle filters. Technical report, Institutionen för Systemteknik, Linköpings Univ., May 2004.

[10] K. Kanazawa, D. Koller, and S. J. Russell. Stochastic simulations algorithms for dynamic probabilistic networks. In *Proc. of the European Conference on Computer Vision*, 1996.

[11] J. MacCormickn and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proc. of 7th International Conference on Computer Vision (ICCV)*, pages 572–587, 1999.

[12] D. Schulz, D. Fox, W. Burgard, and A. B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. In *International Journal of Robotics Research (IJRR)*, 2003.