A Joint Probability Data Association Filter Algorithm for Multiple Robot Tracking Problems

Aliakbar Gorji Daronkolaei, Vahid Nazari, Mohammad
Bagher Menhaj, and Saeed Shiry
Amirkabir University of Technology, Tehran,
Iran

1. Introduction

Estimating the position of a mobile robot in a real environment is taken into account as one of the most challenging topics in the recent literature (Fox et al., 2000). This problem can be usually explored in two ways. Firstly, a mobile robot should be able to have knowledge about its current position. The Dead-reckoning of a mobile robot may be used to update the position of the robot assuming the initial position is known. However, the encoders of a robot cannot provide precise measurements and, therefore, the position obtained by this way is not reliable. To achieve more accurate approximation of a robot's position, measurements obtained by sensors set on a robot are used to correct the information provided by the encoders. If the mapping of a physical environment is known, the above-mentioned procedure can be easily accomplished by using some well-known approaches such as Kalman filtering (Kalman & Bucy, 1961) to localize the exact position of a mobile robot (Siegwart & Nourbakhsh, 2004). However, when there is not any knowledge about the map, mapping and localization should be conducted simultaneously. The aforementioned topic is known as Simultaneous Localization and Mapping (SLAM) in the literature (Howard, 2005).

In many applications, one may intend to localize other robots' position via a reference robot. Robot soccer problems or people tracking scenarios can be fallen in the pre-mentioned category. Although this problem appears similar to the common localization algorithms, the traditional approaches can not be used because the reference robot does not access to the odometry data of each mobile robot used in localization algorithms to predict the future position of the robot. This issue may be completely perceivable in the people tracking scenario because there is not any information about the movement of people. In this case, some models should be proposed to represent the movement of each object. By defining a suitable motion model for each target and using measurements provided by a reference robot about the current position of the moving object, a linear/nonlinear state space model is constructed representing the movement of each object.

The above-discussed topic can be fallen in the category of target tracking problems where the final aim is defined as tracking the position of a mobile object by a reference sensor. Because of inaccurate data obtained by sensors and uncertain motion models which may not provide reliable prediction of an object's movement, filtering algorithms are used to extract

the position of a mobile target. Kalman filtering has been the first method applied to the field of target tracking. However, the Kalman method and, even, its generalized form known as extended Kalman filter (EKF) (Anderson & Moore, 1979) do not provide reliable results for nonlinear state space models. This problem is very common in tracking applications where the sensor algebraic equations are usually nonlinear towards the position of a target. To remedy the above problems, nonlinear filtering using the particle filter algorithm has been proposed (Gordon et al., 1993), (Doucet et al., 2001). Particle filtering has been extensively applied to many real themes such as aircraft tracking (Ristic et al., 2004), target detection/tracking (Ng et al., 2004), navigation (Gustafsson et al., 2002), training artificial neural networks (Freitas 1999), control (Andrieu et al., 2004), etc. Besides the ease of use, particle filter algorithms lead to much more accurate results than kalman based approaches. Recently, the combination of particle and Kalman filtering has been also applied to many tracking applications, specially, when some parameters of a motion model may be also estimated beside the position of a target (Sarkka et al., 2005).

Multiple robot tracking is another attractive issue in the field of mobile robotics. This area can be imagined as a generalized type of the common tracking problem. In other words, a reference robot should localize other robots/agents position based on information obtained by sensors. To do so, measurements should be associated to the appropriate target. Moreover, some measurements may have been received from unwanted targets usually known as clutters. The combination of the data association concept and common filtering approaches has been used in the literature as the joint probability data association filter (JPDAF) algorithm (Vermaak et al., 2005). This algorithm has been greatly used in many applications such as multiple target tracking (Li et al., 2007), (Fortman et al., 1983), people tracking (Schulz et al., 2003), and security planning (Oh et al., 2004). However, in the field of multiple robot tracking, no comprehensive work has been done and many problems are yet open. For example, unlike the traditional multiple target tracking scenarios in which sensors may be assumed to be fixed or conduct an independent movement, a reference robot can make an organized movement to track other robots position much more precisely. In other words, the motion of a reference robot must be planned so that the robot can track other robots much better. Although this case has been discussed in the literature as observer trajectory planning (OTP) (Singh et al., 2007), proposed approaches are usually implemented in an offline mode. This problem may not be so desirable in multiple robot tracking scenarios where a reference robot should localize other robots simultaneously.

In this paper, some improvements are made on the traditional JPDAF algorithm for multiple robot tracking applications. To provide a better representation of a robot's movement, different motion models proposed in the literature are used to evaluate the efficiency of tracking. Moreover, a new fuzzy controller is proposed to find an optimal trajectory for the movement of the reference robot. It will be shown that this fuzzy controller minimizes the sum of distances between the reference robot and other mobile objects.

To maintain all of above-mentioned topics, this paper is organized as follows. Section 1 deals with the general theory of the JPDAF algorithm. The particle filter algorithm and the concept of data association will be covered in this section. Section 3 discusses the JPDAF algorithm for multiple robot tracking. In this section of the paper, different motion models describing the movement of a mobile robot are represented. Section 4 is devoted to present fuzzy logic controller for optimal observer trajectory planning. This section proposes a fuzzy controller which can join with the JPDAF algorithm to enhance the quality of tracking.

Simulation results confirming the superiority of our proposed algorithm are provided in section 6. Finally, section 7 concludes the paper.

2. The JPDAF algorithm for multiple target tracking

In this section, the JPDAF algorithm considered as the most widely and successful strategy for multi-target tracking under data association uncertainty is presented. The Monte Carlo version of the JPDAF algorithm uses the common particle filter approach to estimate the posterior density function of states given measurements $p(\mathbf{x}_t | \mathbf{y}_{1:t})$. Now, consider the problem of tracking N objects. $\mathbf{x}_t^k$ denotes the state of these objects at time t where k=1,2,..,N is the target number. Furthermore, the movement of each target can be described in a general nonlinear state space model as follows:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) + \mathbf{v}_t \tag{1-1}$$

$$\mathbf{y}_t = \mathbf{g}(\mathbf{x}_t) + \mathbf{w}_t \tag{1-2}$$

where $\mathbf{v}_t$ and $\mathbf{w}_t$ are white noises with the covariance matrixes Q and R, respectively. Also, in the above equation, $\mathbf{f}$ and $\mathbf{g}$ are the general nonlinear functions representing the dynamical behavior of the target and the sensor model. The aim of the JPDAF algorithm is to update the marginal filtering distribution for each target $p(\mathbf{x}_t^k | \mathbf{y}_{1:t})$, k=1,2,..,N, instead of computing the joint filtering distribution $p(\mathbf{X}_t | \mathbf{y}_{1:t})$, $\mathbf{X}_t = \mathbf{x}_t^1,...,\mathbf{x}_t^N$. To compute the above distribution function, some remarks should be first noted as

1. That how to assign each target state to a measurement is crucial. Indeed, at each time step the sensor provides a set of measurements. The source of these measurements can be the targets or the disturbances also known as clutters. Therefore, a special procedure is needed to assign each target to its associated measurement. This procedure is designated as Data Association considered as a key stage of the JPDAF algorithm which is described in next sections.
2. Because the JPDAF algorithm updates the estimated states sequentially, a recursive solution should be applied to update the states at each sample time. Traditionally, Kalman filtering has been a strong tool for recursively estimating the states of the targets in the multi-target tracking scenario. Recently, particle filters joint with the data association strategy have provided better estimations, specially, when the sensor model is nonlinear.

With regard to the above points, the following sections describe how particle filters paralleled with the data association concept can deal with the multi-target tracking problem.

2.1 The particle filter for online state estimation

Consider the problem of online state estimation as computing the posterior probability density function $p(\mathbf{x}_t^k | \mathbf{y}_{1:t})$. To provide a recursive formulation for computing the above density function, the following stages are presented:

1. *Prediction stage*: the prediction step is proceeded independently for each target as

$$p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t-1}) = \int\limits_{\mathbf{x}_{t-1}^k} p(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^k) p(\mathbf{x}_{t-1}^k \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}^k \tag{2}$$

2.   *Update stage*: this step can be also described as follows:

$$p(\mathbf{x}_t^k \mid \mathbf{y}_{1t}) \propto p(\mathbf{y}_t \mid \mathbf{x}_t^k) p(\mathbf{x}_t^k \mid \mathbf{y}_{1t-1}) \tag{3}$$

The particle filter algorithm estimates the probability distribution density function $p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t-1})$ by sampling from a specific distribution function as follows:

$$p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t-1}) = \sum_{i=1}^{N} \widetilde{w}_t^i \delta(\mathbf{x}_t - \mathbf{x}_t^i) \tag{4}$$

Here i=1,2,...,N is the sample number, $\widetilde{\mathbf{w}}_t$ is the normalized importance weight and $\delta(.)$ is the delta dirac function. In the above equation, the state $\mathbf{x}_t^i$ is sampled from the proposal density function $q(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^k, \mathbf{y}_{1:t})$. By substituting the above equation in (2) and the fact that states are drawn from the proposal function q, the recursive equation for the prediction step can be written as follows:

$$\alpha_t^i = \alpha_{t-1}^i \frac{p(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^{k,i})}{q(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^{k,i}, \mathbf{y}_{1:t})} \tag{5}$$

where $\mathbf{x}_{t-1}^{k,i}$ is the ith sample of $\mathbf{x}_{t-1}^k$. Now, by using (3) the update stage can be expressed as a recursive adjustment of importance weights as follows:

$$\mathbf{w}_t^i = \alpha_t^i p(\mathbf{y}_t \mid \mathbf{x}_t^k) \tag{6}$$

By repeating the above procedure at each time step, the sequential importance sampling (SIS) algorithm for online state estimation is presented as below:

---

1.   For i=1: N initialize the states $\mathbf{x}_0^i$, prediction weights α_0^i and importance weights w_0^i.
2.   At each time step t proceed the following stages:
     a.   Sample states from the proposal density function as follows:

$$\mathbf{x}_t^i \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^i, \mathbf{y}_{1:t}) \tag{7}$$

     b.   Update prediction weights by (5).
     c.   Update importance weights by (6).
     d.   Normalize importance weights as follows:

$$\widetilde{w}_t^i = \frac{w_t^i}{\sum_{i=1}^{N} w_t^i} \tag{8}$$

3.   Set t=t+1 and go to 2.

Table. 1. The SIS Algorithm

For the sake of simplicity, the index k has been eliminated in Table 1. The main failure of the SIS algorithm is the degeneracy problem. That is, after a few iterations one of the normalized importance ratios tends to 1, while the remaining ratios tend to zero. This problem causes the variance of the importance weights to increase stochastically over time (Del Moral et al., 2006). To avoid the degeneracy of the SIS algorithm, a selection (resampling) stage may be used to eliminate samples with low importance weights and multiply samples with high importance weights. There are many approaches to implement the resampling stage (Del Moral et al., 2006). Among them, the residual resampling provides a straightforward strategy to solve the degeneracy problem in the SIS algorithm. By combining the concept of residual resampling with the SIS algorithm presented before, the SIR algorithm is described in Table 2.

---

1. For i=1: N initialize the states $\mathbf{x}_0^i$, prediction weights α_0^i and importance weights w_0^i.

2. At each time step t do the following stages:

    a. Do the SIS algorithm to sample states $\mathbf{x}_t^i$ and compute normalized importance weights $\widetilde{w}_t^i$.

    b. Check the resampling criterion:

        i. If $N_{eff} > thresh$, follow the SIS algorithm Else:

        ii. Implement the residual resampling stage to multiply/suppress $\mathbf{x}_t^i$ with high/low importance weights.

        iii. Set the new normalized importance weights as $\widetilde{w}_t^i = \dfrac{1}{N}$.

3. Set t=t+1 and go to 2.

Table. 2. The SIR Algorithm

In the above algorithm, N_{eff} is a criterion checking the degeneracy problem which can be written as:

$$N_{eff} = \frac{1}{\displaystyle\sum_{i=1}^{N} (\widetilde{w}_t^i)^2} \tag{9}$$

In (Freitas 1999), a comprehensive discussion has been made on how one can implement the residual resampling stage.

Besides the SIR algorithm, some other approaches have been proposed in the literature to enhance the quality of the SIR algorithm such as Markov Chain Monte Carlo particle filters, Hybrid SIR and auxiliary particle filters (Freitas 1999). Although these methods are more accurate than the common SIR algorithm, some other problems such as the computational cost are the most significant reasons that, in many real time applications such as online tracking, the traditional SIR algorithm is applied to recursive state estimation.

2.2 Data association

In the last section, the SIR algorithm was presented for online state estimation. However, the major problem of the proposed algorithm is how to compute the likelihood

function $p(\mathbf{y}_t | \mathbf{x}_t^k)$. To do so, an association should be made between measurements and targets. Generally, two types of association may be defined as follows:

Definition 1: we will denote a target to measurement association ($T \rightarrow M$) by $\tilde{\lambda} = \{\tilde{r}, m_c, m_T\}$ where $\tilde{r} = \{\tilde{r}_1, ..., \tilde{r}_K\}$ and $\tilde{r}_k$ is defined as:

$$\tilde{r}_k = \begin{cases} 0 & \text{If the } k^{th} \text{ t arg } et \text{ is not det } ected \\ j & \text{f the } k^{th} \text{ t arg } et \text{ has generated the } j^{th} \text{ measurement} \end{cases} \tag{10}$$

where j=1,2,...,m and m is the number of measurements at each time step and k=1,2,..,K which K is the number of targets.

Definition 2: in a similar fashion, the measurement to target association ($M \rightarrow T$) is defined as $\lambda = \{r, m_c, m_T\}$ where $r = \{r_1, ..., r_m\}$ and r_j is defined as follows:

$$\tilde{r}_j = \begin{cases} 0 & \text{If the } j^{th} \text{ measurement is not associated} \\ k & \text{if the } j^{th} \text{ measurement is associated to the } k^{th} \text{ t arg } et \end{cases} \tag{11}$$

In both above equations, m_T is the number of measurements due to the targets and m_c is the number of clutters. It is very easy to show that both definitions are equivalent but the dimension of the target to measurement association is less than the measurement to target association dimension. Therefore, in this paper, the target to measurement association is used. Now, the likelihood function for each target can be written as

$$p(\mathbf{y}_t | \mathbf{x}_t^k) = \beta^0 + \sum_{i=1}^{m} \beta^i p(\mathbf{y}_t^i | \mathbf{x}_t^k) \tag{12}$$

In the above equation, β^i is defined as the probability that the ith measurement is assigned to the kth target. Therefore, β^i can be written as the following equation:

$$\beta^i = p(\tilde{r}_k = i | \mathbf{y}_{1:t}) \tag{13}$$

Before describing how to compute the above equation, the following definition is represented:

Definition 3: we define the set $\tilde{\lambda}$ as all possible assignments which can be made between measurements and targets. For example, consider a 3-target tracking problem. Assume that the following associations are recognized between targets and measurements:

$r_1 = \{1, 2\}$, $r_2 = \{3\}$, $r_3 = \{0\}$. Now, the set $\tilde{\lambda}$ can be shown in Table 3.

Target 1	Target 2	Target 3
0	0	0
0	3	0
1	0	0
1	3	0
2	0	0
2	3	0

Table. 3. All possible associations between the targets and measurements for example 1

In Table 3, 0 means that the target has not been detected. By using the above definition, (11) can be rewritten as

$$p(\widetilde{r}_k = i \mid \mathbf{y}_{1:t}) = \sum_{\widetilde{\lambda}_t \in \Lambda_t, \widetilde{r}_k = j} p(\widetilde{\lambda}_t \mid \mathbf{y}_{1:t}) \tag{14}$$

The right side of the above equation is written as follows:

$$p(\widetilde{\lambda}_t \mid \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \widetilde{\lambda}_t) p(\mathbf{y}_{1:t-1}, \widetilde{\lambda}_t)}{p(\mathbf{y}_{1:t})} = p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \widetilde{\lambda}_t) p(\widetilde{\lambda}_t) \tag{15}$$

In the above equation, we have used the fact that the association vector $\widetilde{\lambda}_t$ is not dependent on the history of measurements. Each of density functions in the right hand side of the above equation can be computed as follows:

1. $p(\widetilde{\lambda}_t)$

The above density function can be written as

$$p(\widetilde{\lambda}_t) = p(\widetilde{r}_t, m_c, m_T) = p(\widetilde{r}_t \mid m_c, m_T) p(m_c) p(m_T) \tag{16}$$

The computation of each density function in the right hand side of the above equation is straightforward and can be found in (Freitas 1999).

2. $p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \widetilde{\lambda}_t)$

Because targets are assumed to be independent, the above density function can be written as follows:

$$p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \widetilde{\lambda}_t) = (V_{\max})^{-m_c} \Pi_{j \in \Gamma} p^{r_t^j}(\mathbf{y}_t^j \mid \mathbf{y}_{1:t-1}) \tag{16}$$

where $V_{\max}$ is the maximum volume in the line of sight of sensors, Γ is the set of all valid measurement to target associations and $p^{r_t^j}(\mathbf{y}_t^j \mid \mathbf{y}_{1:t-1})$ is the predictive likelihood for the $(r_t^j)^{th}$ target. Now, consider $r_t^j = k$. The predictive likelihood function for the kth target can be written as follows:

$$p^k(\mathbf{y}_t^j \mid \mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t^j \mid \mathbf{x}_t^k) p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_t^k \tag{17}$$

Both density functions in the right hand side of the above equation are estimated using the samples drawn from the proposal density function. However, the main problem is how to determine the association between measurements $\mathbf{y}_t$ and targets. To do so, the soft gating method is proposed in Table 4.

1. Consider $\mathbf{x}_t^{i,k}$, i=1,2,..,N, as the samples drawn from the proposal density function.

2. For j=1:m do the following steps for the jth measurement:

    a. For k=1:K do the following steps:

        i. Compute μ_j^k as follows:

$$\mu_j^k = \sum_{i=1}^{N} \widetilde{\alpha}_t^{i,k} g(\mathbf{x}_t^{i,k}) \tag{18}$$

   Where g is the sensor model and $\widetilde{\alpha}_t^k$ is the normalized weight as presented in the SIR algorithm.

        ii. Compute σ_j^k by the following equation:

$$\sigma_j^k = R + \sum_{i=1}^{N} \widetilde{\alpha}_t^{i,k} \Big(g(\mathbf{x}_t^{i,k}) - \mu_j^k\Big)\Big(g(\mathbf{x}_t^{i,k}) - \mu_j^k\Big)^T \tag{19}$$

        iii. Compute the distance to the jth target as follows:

$$d_j^2 = \frac{1}{2}(\mathbf{y}_t^j - \mu_j^k)^T (\sigma_j^k)^{-1} (\mathbf{y}_t^j - \mu_j^k) \tag{20}$$

        iv. If $d_j^2 < \varepsilon$, assign the jth measurement to the kth target.

    b. End of the loop for k.

3. End of the loop for j.

Table. 4. Soft gating for data association

It is easy to show that the predictive likelihood function presented in (16) can be approximated as follows:

$$p^k(\mathbf{y}_t^j \mid \mathbf{y}_{1:t-1}) \approx N(\mu^k, \sigma^k) \tag{21}$$

where $N(\mu^k, \sigma^k)$ is a normal distribution with mean μ^k and the covariance matrix σ^k. By computing the predictive likelihood function, the likelihood density function can be easily estimated. In the next subsection, the JPDAF algorithm is presented for multi-target tracking.

2.3 The JPDAF algorithm

The mathematical foundations of the JPDAF algorithm were discussed in the last sections. Now, we are ready to propose the full JPDAF algorithm for the problem of multiple target tracking. To do so, each target is characterized by a dynamical model introduced by (1). The JPDAF algorithm is, therefore, presented in Table 5.

1. **Initialization:** initialize states for each target as $\mathbf{x}_0^{i,k}$ where i=1,2,..,N and k=1,2,...,K, the predictive importance weights $\alpha_0^{i,k}$ importance weights $w_0^{i,k}$.

2. At each time step t proceed through the following stages:
   a. For k=1:K conduct the following procedures for each target:
      i. For i=1:N do the following steps:
      ii. Sample the new states from the proposal density function as follows:

      $$\mathbf{x}_t^{i,k} \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{i,k}, \mathbf{y}_{1:t}) \tag{22}$$

      iii. Update the predictive importance weights as

      $$\alpha_t^{i,k} = \alpha_{t-1}^{i,k} \frac{p(\mathbf{x}_t^{i,k} \mid \mathbf{x}_{t-1}^{i,k})}{q(\mathbf{x}_t^{i,k} \mid \mathbf{x}_{t-1}^{i,k}, \mathbf{y}_{1:t})} \tag{23}$$

      Then, normalize the predictive importance weights.

      iv. Use the sampled states and new observations $\mathbf{y}_t$ to constitute the association vector for each target as $R^k = \{j \mid 0 \leq j \leq m, \mathbf{y}_j \rightarrow k\}$ where $(\rightarrow k)$ refers to the association between the kth target and the jth measurement. Use the soft gating procedure described in the last subsection to compute each association.

      v. Constitute all possible associations for the targets and make the set $\tilde{\Gamma}$ as described in Definition 3.

      vi. Use (13) and compute β^l for each measurement where l=1,2,..,m and m is the number of measurements.

      vii. By using (10) compute the likelihood ratio for each target as $p(\mathbf{y}_t \mid \mathbf{x}_t^{i,k})$.

      viii. Compute importance weights and normalize them as follows:

      $$w_t^{i,k} = \alpha_t^{i,k} p(\mathbf{y}_t \mid \mathbf{x}_t^{i,k}), \quad \tilde{w}_t^{i,k} = \frac{w_t^{j,k}}{\displaystyle\sum_{i=1}^{N} w_t^{j,k}} \tag{24}$$

      ix. Implement the resampling stage. To do so, do the similar procedure described in the SIR algorithm. Afterwards, for each target the resampled states can be presented as follows:

      $$\{\tilde{w}_t^{i,k}, \mathbf{x}_t^{i,k}\} \rightarrow \{\frac{1}{N}, \mathbf{x}_t^{m(i),k}\} \tag{25}$$

   b. End of the loop for k.
3. Set t=t+1 and go to step 2.

Table. 5. The JPDAF algorithm for multiple target tracking

The above algorithm can be used to deal with the multi-target tracking scenario. In the next section, we show how the JPDAF algorithm can be used for multiple robot tracking. In addition, some well-known motion models are presented to track the motion of a mobile robot.

3. The JPDAF algorithm for multiple robot tracking

In the last section, the JPDAF algorithm was completely discussed. Now, we want to use the JPDAF algorithm for the problem of multiple robot tracking. To do so, consider a simple 2-wheel differential mobile robot, Fig. 1, whose dynamical model is represented as follows:
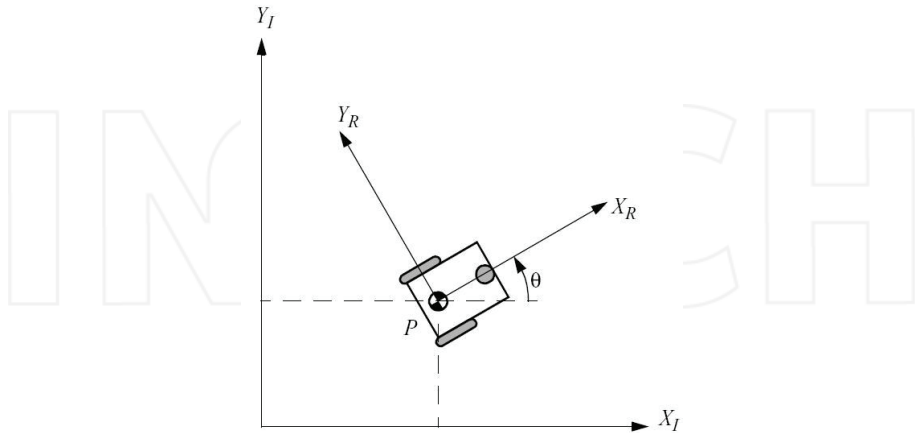

Fig. 1. A 2-wheel differential mobile robot

$$x_{t+1} = x_t + \frac{\Delta s_r + \Delta s_l}{2} \cos(\theta_t + \frac{\Delta s_r - \Delta s_l}{2b})$$

$$y_{t+1} = y_t + \frac{\Delta s_r + \Delta s_l}{2} \sin(\theta_t + \frac{\Delta s_r - \Delta s_l}{2b})$$

$$\theta_{t+1} = \theta_t + \frac{\Delta s_r - \Delta s_l}{b} \tag{26}$$

where $[x_t, y_t]$ is the position of the robot, θ_t is the angle of the robot's head, Δs_r and Δs_l are the distances travelled by each wheel, and b refers to the distance between two wheels of the robot. The above equation describes a simple model presenting the motion of a differential mobile robot. For a single mobile robot localization problem, the most straightforward way is to use this model and the data collected from the sensors set on the left and right wheels measuring Δs_r and Δs_l at each time step. But, the above method does not lead to the suitable results because the data gathered from the sensors are dotted with the additive noise and, therefore, the estimated trajectory does not match the actual trajectory. To remedy this problem, measurements obtained from the sensors are used to modify the estimated states. Therefore, Kalman and particle filters have been greatly applied to the problem of mobile robot localization (Siegwart & Nourbakhsh, 2004). Now, consider the case in which the position of other robots should be identified by a reference robot. In this situation, the dynamical model discussed previously is not applicable because the reference robot does not have any access to the internal sensors of other robots such as the sensors

measuring the movement of each wheel. Therefore, a motion model should be first defined for the movement of each mobile robot. The simplest model is a near constant velocity model presented as follows:

$$\mathbf{X}_{t+1} = A\mathbf{X}_t + B\mathbf{u}_t$$
$$\mathbf{X}_t = [x_t, \dot{x}_t, y_t, \dot{y}_t]$$

(27)

where the system matrixes are defined as

$$A = \begin{pmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} \dfrac{T_s^2}{2} & 0 \\ T_s & 0 \\ 0 & \dfrac{T_s^2}{2} \\ 0 & T_s \end{pmatrix}$$

(28)

where T_s refers to the sample time. In the above equations, $\mathbf{u}_t$ is a white noise with zero mean and an arbitrary covariance matrix. Because the model is supposed to be a near constant velocity model, the covariance of the additive noise should not be so large. Indeed, this model is suitable for the movement of targets with a constant velocity which is common in many applications such as aircraft path tracking and people tracking.

The movement of a mobile robot can be described by the above model in many conditions. However, in some special cases the robots' movement can not be characterized by a simple near constant velocity model. For example, in a robot soccer problem, the robots may conduct a manoeuvring movement to reach a special target such as a ball. In these cases, the robot changes its orientation by varying input forces imposed to the right and left wheels. Therefore, the motion trajectory of the robot is so complex that an elementary constant velocity model may not result in satisfactory responses. To overcome the mentioned problem, the variable velocity model is proposed. The key idea behind this model is using the robot's acceleration as another state variable which should be estimated as well as the velocity and position of the robot. Therefore, the new state vector is defined as $\mathbf{X}_t = [x_t, \dot{x}_t, a_t^x, y_t, \dot{y}_t, a_t^y]$ where a_t^x, a_t^y are the robot's acceleration along with the x and y axis, respectively. Now, the near constant acceleration model can be written similar to what mentioned in (27), except for the system matrixes which are defined as follows (Ikoma et all., 2001):

$$A = \begin{pmatrix} 1 & 0 & T_s & 0 & a_1 & 0 \\ 0 & 1 & 0 & T_s & 0 & a_1 \\ 0 & 0 & 1 & 0 & a_2 & 0 \\ 0 & 0 & 0 & 1 & 0 & a_2 \\ 0 & 0 & 0 & 0 & \exp(-T_s) & 0 \\ 0 & 0 & 0 & 0 & 0 & \exp(-T_s) \end{pmatrix}, B = \begin{pmatrix} b_1 & 0 \\ 0 & b_1 \\ b_2 & 0 \\ 0 & b_2 \\ b_3 & 0 \\ 0 & b_3 \end{pmatrix}$$

(30)

where the constants of the above equation are defined as

$$b_3 = \frac{1}{c}(1 - \exp(-cT_s)), \, a_2 = b_3, \, b_2 = \frac{1}{c}(T_s - a_2)$$

$$a_1 = b_2, \, b_1 = \frac{1}{c}\left(\frac{T_s^2}{2} - a_1\right)$$

(31)

In the above equation, c is a constant value. The above model can be used to track the motion trajectory of a manoeuvring object, such as the movement of a mobile robot. Moreover, using the results of the proposed models can enhance the performance of tracking. This idea can be presented as follows:

$$\widetilde{\mathbf{X}}_t = \alpha \widetilde{\mathbf{X}}_t^a + (1 - \alpha)\widetilde{\mathbf{X}}_t^v$$

(32)

where $\widetilde{\mathbf{X}}_t^a$ and $\widetilde{\mathbf{X}}_t^v$ are the estimation of the near constant acceleration and near constant velocity model, respectively, and α is a number between 0 and 1. To adjust α, an adaptive method is used considering the number of measurements assigned to targets when each model is used separately. That is, more the number of measurements is assigned to targets by each model, the larger value is chosen for α.

Besides the above approaches, some other methods have been proposed to deal with tracking of manoeuvring objects such as Interactive Multiple Mode (IMM) filters (Pitre et al., 2005). However, these methods are applied to track the movement of objects with sudden manoeuvring movement which is common in aircrafts. In the mobile robots scenario, the robot's motion is not so erratic that the above method is necessary. Therefore, the near constant velocity model, near constant acceleration model or a combination of the proposed models can be considered as the most suitable structures which can be used to represent the dynamical model of a mobile robot. Afterwards, the JPDAF algorithm can be easily applied to the multi-robot tracking problem.

4. A fuzzy controller for optimal observer trajectory planning

In this section, a novel fuzzy logic controller (FLC) is proposed to maintain a better tracking quality for the multi-robot tracking problem. Indeed, the major motivation of using a moving platform for the reference robot is some weaknesses in the traditional multiple target tracking scenarios in which sensors were assumed to be fixed or conduct an independent movement from mobile targets. The most important weaknesses found in recent approaches are:

- Generally, the variance of the additive noise of sensors increases when the distance between each sensor and mobile targets increases. Consequently, the quality of tracking will decrease if the position of the sensor or reference robot is fixed.
- If the position of the reference robot is assumed to be fixed, targets may exit the reference robot's field of view and, therefore, the reference robot will lose other robots' position.
- In some applications, the reference robot may be required to track a special target. This problem is common in some topics such as the robot rescue or security planning, when the reference robot should follow the movement of an individual. The traditional approaches are not flexible enough to be applied for the path/trajectory following problem.

- Switching between trajectory tracking and path following is either impossible or very hard and time consuming when traditional approaches are used, when the reference robot is fixed or its movement is independent from other robots' movement.

To remedy aforementioned flaws, a novel strategy is proposed to provide a trajectory for the reference robot dependent on other targets' movement. To obtain an optimal solution, the following cost function is defined for the trajectory planning problem:

$$R = \sqrt{\frac{\sum\limits_{i=1}^{n}(r_i)^2}{n}}$$

$$\Phi = \sqrt{\frac{\sum\limits_{i=1}^{n}(\varphi_i)^2}{n}}$$

(33)

where r_i is the distance between the ith target and reference robot and φ_i is the angle between the ith target and reference robot in the local coordination of the reference robot (Fig. 2).
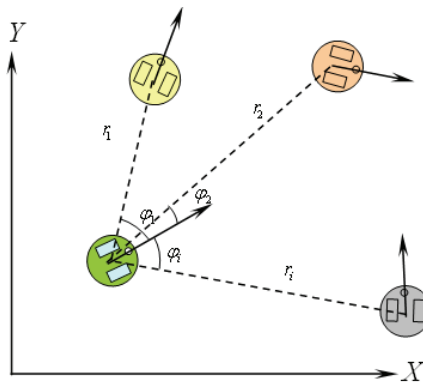
Fig. 2. Position of targets in the reference robot's coordination

The major reason for defining the above cost function is that, by this way, the reference robot maintains its minimal distance to all targets and, thus, the performance of the target tracking is improved. In other words, when the reference robot is placed in a position whose sum of distances to other robots is minimal, the effect of the additive noise dependent on the distance between targets and the reference robot is also minimized. Therefore, the trajectory planning problem is defined as a method causing the reference robot to move along a suitable path minimizing the pre-mentioned cost function.

In order to execute the aforementioned trajectory planning problem, a robust fuzzy controller is used. Fig. 3 shows the block diagram of the proposed fuzzy controller. In design of fuzzy logic controllers, we use the Mamdani type of the fuzzy control containing fuzzification and defuzzification stages and, also, a rule base. The task of the fuzzy controller is to have the reference robot follow the above-discussed optimal trajectory smoothly and, of course, precisely. In this paper, we use an FLC based on kinematical model of the robot.
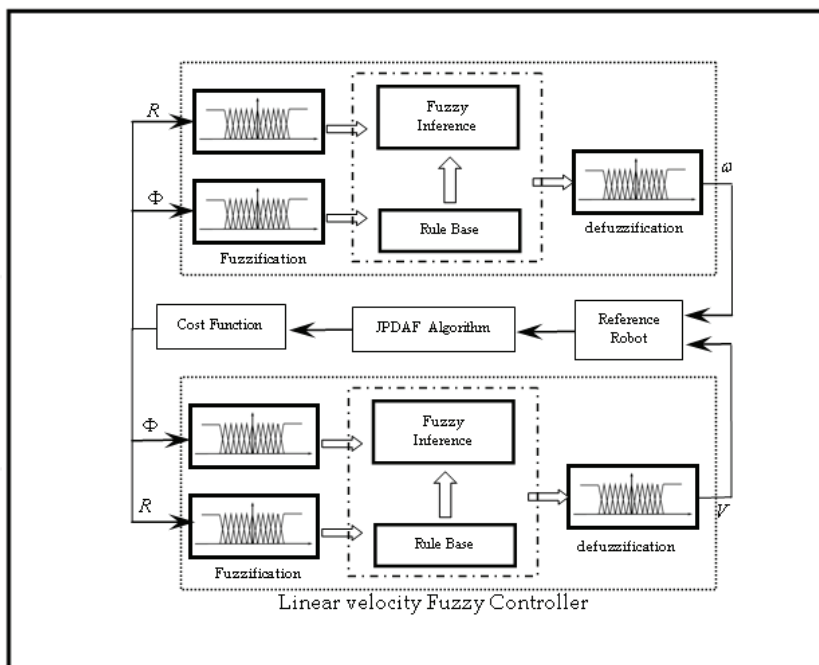
Fig. 3. A block diagram of represented fuzzy controller

After testing various number of membership functions for input variables R, Φ, the best fuzzy system for the angular velocity control is designed with *seven* triangular membership functions. Although there is no restriction on the general form of membership functions, we choose the piecewise linear description (Fig. 4 & Fig. 5).
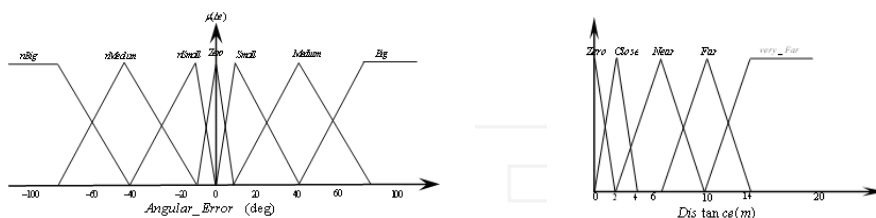

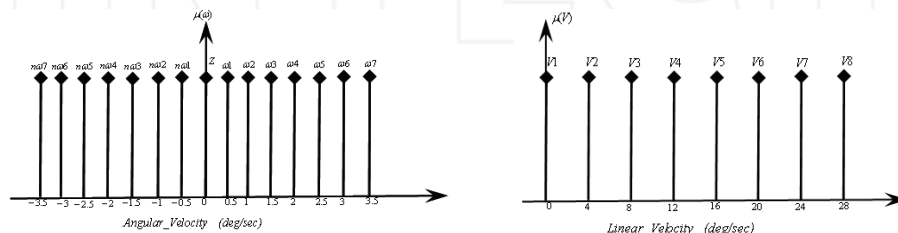Fig. 4. Membership functions of input fuzzy sets of fuzzy controllers

Fig. 5. Membership functions of output fuzzy sets of fuzzy controllers

The second step in designing an FLC is the fuzzy inference mechanism. The knowledge base of the angular and linear velocity of fuzzy controllers consists of the rules described in Table 6 & Table 7.

Big	Medium	Small	Zero	nSmall	nMedium	nBig	Φ R
$\Omega 7$	$\omega 6$	$\Omega 5$	Z	$n\omega 5$	$n\omega 6$	$n\omega 7$	**Zero**
$\Omega 6$	$\omega 5$	$\Omega 4$	Z	$n\omega 4$	$n\omega 5$	$n\omega 6$	**Close**
$\Omega 5$	$\omega 4$	$\Omega 3$	Z	$n\omega 3$	$n\omega 4$	$n\omega 5$	**Near**
$\Omega 4$	$\omega 3$	$\Omega 2$	Z	$n\omega 2$	$n\omega 3$	$n\omega 4$	**Far**
$\Omega 3$	$\omega 2$	$\Omega 1$	Z	$n\omega 1$	$n\omega 2$	$n\omega 3$	**very_Far**

Table. 6. The fuzzy rule base for the angular velocity fuzzy controller

Big	Medium	Small	Zero	nSmall	nMedium	nBig	Φ R
V1	V2	V3	V4	V3	V2	V1	**Zero**
V2	V3	V4	V5	V4	V3	V2	**Close**
V3	V4	V5	V6	V5	V4	V3	**Near**
V4	V5	V6	V7	V6	V5	V4	**Far**
V5	V6	V7	V8	V7	V6	V5	**very_Far**

Table. 7. The fuzzy rule base for the linear velocity fuzzy controller

In this application, an algebraic product is used for all of the *t-norm* operators, max is used for all of the *s-norm*, as well as individual-rule based inference with union combination and mamdani's product implication . Product inference engine is defined as

$$\mu_{B'}(y) = \max_{l=1}^{M} \left[\sup_{x \in U} \left(\mu_{A'}(x) \prod_{i=1}^{n} \mu_{A_i^l}(x_i) \mu_{B^l}(y) \right) \right] \tag{34}$$

There are many alternatives to perform the defuzzification stage. The strategy adopted here is the Center Average defuzzification method. This method is simple and very quick and can be implemented by (Wang 1997)

$$y^* = \frac{\sum_{l=1}^{M} \bar{y}^l w_l}{\sum_{l=1}^{M} w_l} \tag{35}$$

where $\bar{y}^l$ is the center of th l^{th} fuzzy set and w_l is its heigh.

The above procedures provide a strong tool for designing a suitable controller for the reference robot leading to a better tracking performance. The next section shows how this approach enhances the accuaracy of tracking.

5. Simulation results

To evaluate the efficiency of the JPDAF algorithm, a 3- robot tracking scenario is designed. Fig. 6 shows the initial position of the reference and target robots. To prepare the scenario and implement the simulations, parameters are first determined for the mobile robots' structure and simulation environment by Table 8. Now, the following experiments are conducted to evaluate the performance of the JPDAF algorithm in various situations.

| Parameters	Description
v_l	The robot's left wheel speed
v_r	The robot's right wheel speed
b	The distance between the robot's wheels
n_s	Number of sensors placed on each robot
R_{max}	Maximum range of simulation environment
Q_s	The covariance matrix of the sensors' noise
t_s	Sample time for simulation
t_{max}	Maximum running time for the simulation

Table. 8. Simulation parameters
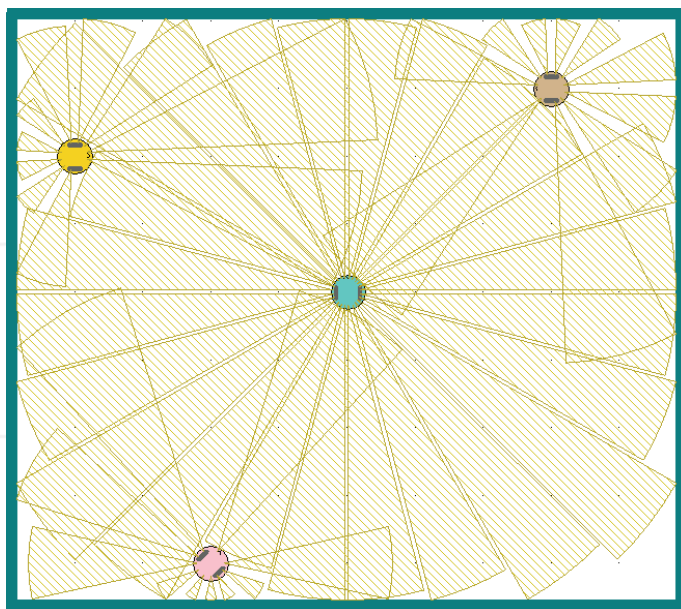
Fig. 6. The generated trajectory for mobile robots in a simulated environment

5.1 Local tracking for manoeuvring movement and the fixed reference robot

To design a suitable scenario, the speed of each robot's wheel is determined by Table 3. The initial position of the reference robot is set in [0, 0, 0]. Moreover, other robots' position is assumed to be [2, 2, π], [1, 3, $\frac{\pi}{3}$], [1, 9, $\frac{\pi}{4}$], respectively. To run the simulation, sample time t_s and maximum running time are also set in 1s and 200s, respectively. To consider the uncertainty in measurements provided by sensors, a Gaussian noise with zero mean and the covariance matrix $\begin{pmatrix} .2 & 0 \\ 0 & 5 \times 10^{-4} \end{pmatrix}$ is added to measurements obtained by sensors. After simulating the above-mentioned 3-robot scenario with aforementioned parameters, the generated trajectories can be observed in Fig. 7.
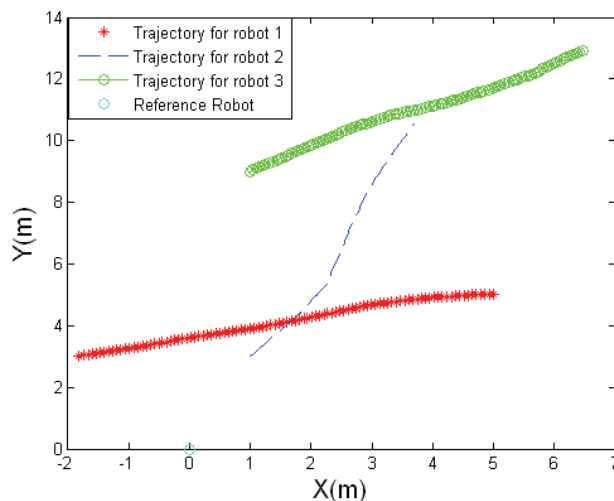
Fig. 7. Generated trajectories for multi-robot tracking scenario

Now, we are ready to implement the JPDAF algorithm discussed before. First, a JPDAF algorithm with 500 particles is used to track the each target's movement. To compare the efficiency of each motion model described in section 3, simulations are done for different motion models. Fig. 8 shows the tracking results for various models and targets. To provide a better view to the accuracy of each approach, Table 9 and Table 10 present the tracking error for each model where the following criterion is used to compute the error:

$$e_z = \frac{\sum_{t=1}^{t_{max}} (z_t - \tilde{z}_t)^2}{t_{max}}, z_t = \{x_t, y_t\} \tag{35}$$

From Tables, it is obvious that the combined model has resulted in a better performance than other motion models. Indeed, near constant velocity and acceleration models do not provide similar results during the simulation and, therefore, the combined model mixing results obtained from each model has led to much better performance.
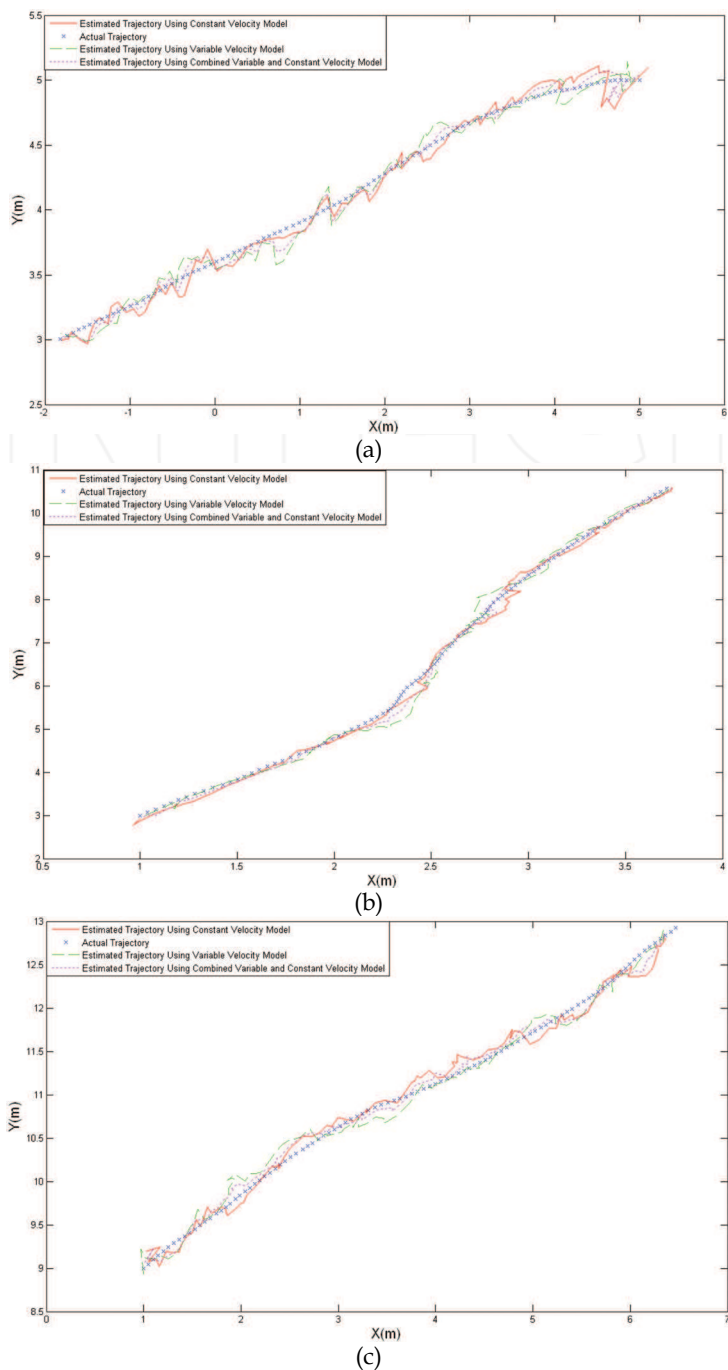
(a)

(b)

(c)

Fig. 8. Tracking results using the JPDAF algorithm with 500 particles for all robots

Robot Number	Const. Velocity	Const. Acceleration	Combined Model
1	0.002329	0.006355	0.002549
2	0.003494	0.003602	0.001691
3	0.005642	0.005585	0.002716

Table. 9. Tracking errors of estimating x_t for various motion models

Robot Number	Const. Velocity	Const. Acceleration	Combined Model
1	0.005834	0.006066	0.003755
2	0.016383	0.009528	0.007546
3	0.011708	0.012067	0.006446

Table. 10. Tracking errors of estimating y_t for various motion models

5.2 Local tracking using a mobile reference robot

Now, we apply the control strategy presented in section 4 for finding an optimal trajectory for the reference robot. To implement the simulation, the reference robot is placed at [0, 0]. A fuzzy controller with two outputs is designed to find the linear and angular velocity of the reference robot. Simulations are conducted with parameters similar to ones defined in the last section. Moreover, to consider the effect of the distance between sensors and targets, the covariance of the additive noise is varied by changing the distance. Fig. 9 explains how we have modelled the uncertainty in measurements received by sensors.

After running the simulation for 400s, the fuzzy controller finds a trajectory for the mobile robot. Fig. 10 shows the obtained trajectory after applying the JPDAF algorithm with 500 particles for finding the position of each mobile target.
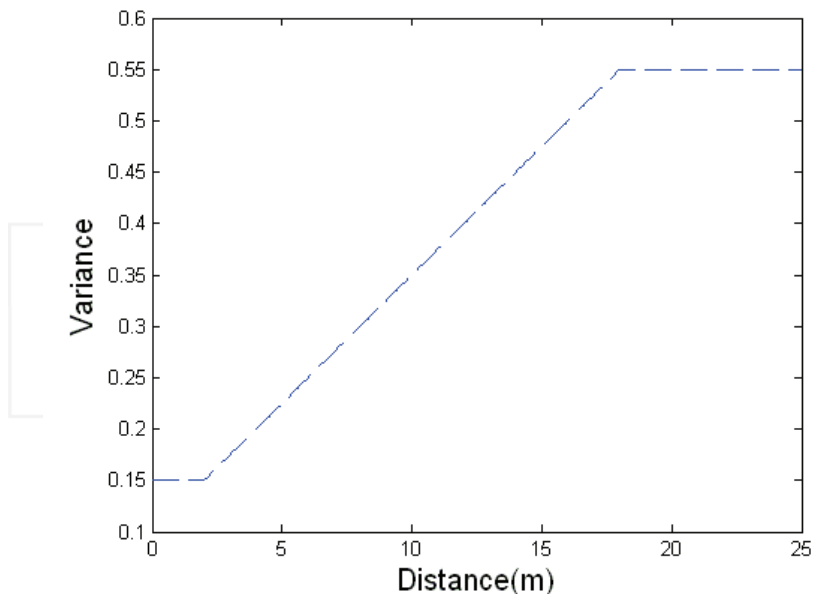

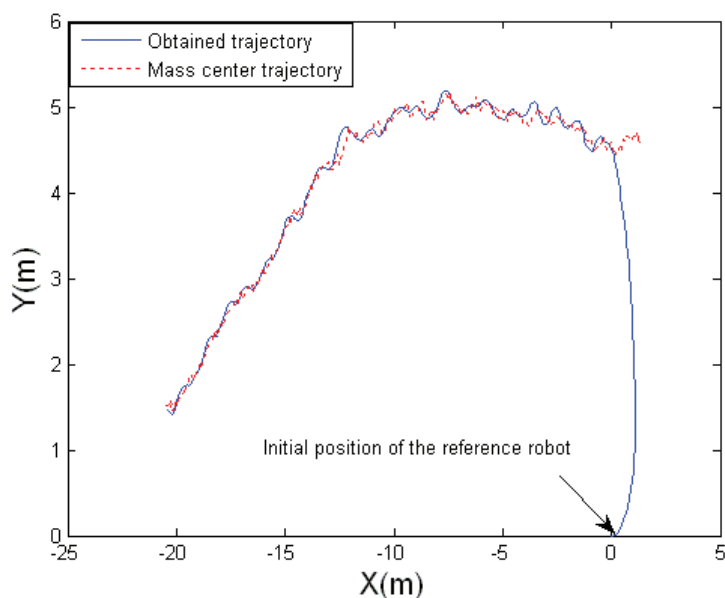Fig. 9. A model used to describe the behaviour of additive noises by changing the distance

Fig. 10. The obtained trajectory for the reference robot using the fuzzy controller

The above figure justifies this fact that the reference robot has been directed to the geometrical placement of the targets' mass centre. Additionally, the simulation is run for situations in which the reference robot is fixed and, also, mobile with an arbitrary trajectory, without applying the control scheme. Fig. 11 shows tracking results using each of the above-mentioned strategies. Also, tables 11 & 12 present the estimated tracking error for all cases. Simulation results show the better tracking performance after applying the control strategy. In other words, because the controller directs the reference robot to a path in which the sum of the distance to other robots is minimal, the tracking algorithm is able to find other robots' position much better than the case in which the reference robot is fixed. In addition, when the reference robot is moving without any special plan, or, at least, its movement is completely independent from other robots' movement, the tracking performance appears much worse. The above-mentioned evidence shows that the effect of the additive noise's variance is so much that the reference robot has completely lost the trajectory of the target shown in Fig. 11 (c). In this case, using the control strategy has caused the reference robot to be placed in a position equally far from other robots and, therefore, the effect of the additive noise weakens. The most important advantage of our proposed approach compared with other methods suggested in the literature for observer trajectory planning (Singh et al., 2007) is that the fuzzy controller can be used in an online mode while recent approaches are more applicable in offline themes. The aforementioned benefit causes our approach to be easily applied to many real topics such as robot rescue, simultaneous localization and mapping (SLAM), etc.
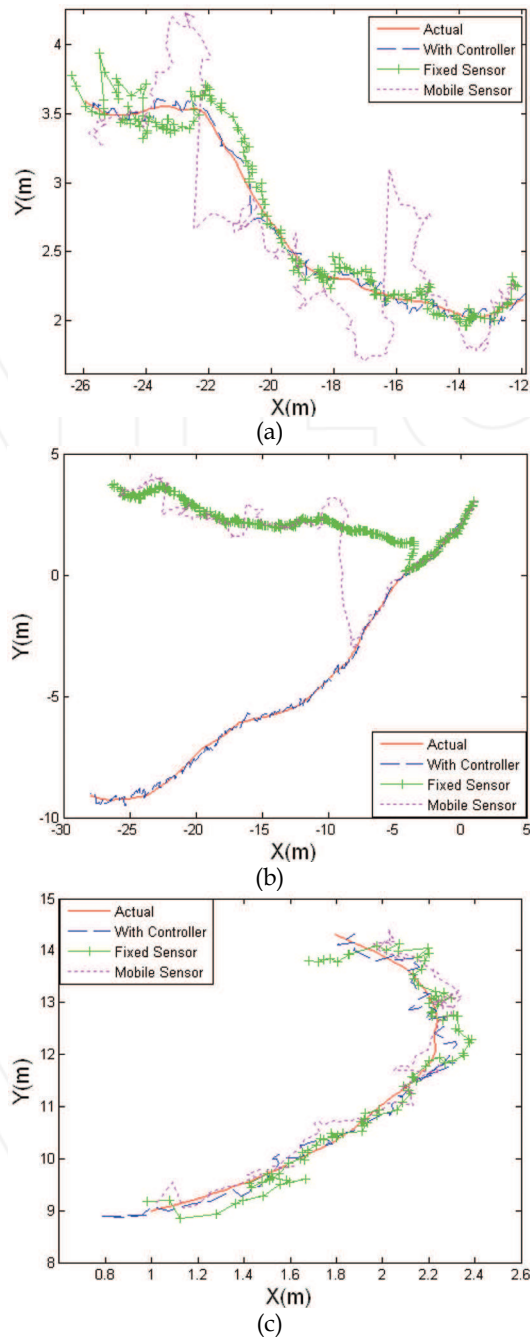
(a)

(b)

(c)

Fig. 11. Tracking results for various situations of the reference robot (fixed, mobile with and without a controller)

Robot Number	Fixed Ref. Robot	Mobile Ref. Robot	Ref. Robot With Controller
1	0.024857	0.077568	0.005146
2	0.883803	0.821963	0.005236
3	0.009525	0.025623	0.009882

Table. 11. Tracking errors of estimating x_t for various situations of the reference robot

Robot Number	Fixed Ref. Robot	Mobile Ref. Robot	Ref. Robot With Controller
1	0.010591	0.02129	0.004189
2	62.54229	57.06329	0.013196
3	0.023677	0.034152	0.018124

Table. 12. Tracking errors of estimating y_t for various situations of the reference robot

6. Conclusion

This paper dealt with the problem of multi-robot tracking taken into account as one of the most important topics in robotics. The JPDAF algorithm was presented for tracking multiple moving objects in a real environment. Then, extending the aforementioned algorithm to robotics application was discussed. To enhance the quality of tracking, different motion models were introduced along with a simple near constant velocity model. Proposing a new approach for observer trajectory planning was the key part of this paper where it was shown that because of some problems such as increasing the variance of the additive noise by increasing the distance between targets and the reference robot, the tracking performance may be corrupted. Therefore, a fuzzy controller was proposed to find an optimal trajectory for the reference robot so that the effect of the additive noise is minimized. Simulation results presented in the paper confirmed the efficiency of the proposed fuzzy control approach in enhancing the quality of tracking.

7. References

Fox, D; Thrun. S; Dellaert. F & Bugard, W (2000). *Particle Filters For Mobile Robot Localization*, Sequential Monte Carlo in Practice, Springer, Verlag.

Kalman, R. E. & Bucy R. S. (1961). *New Results in Linear Filtering and Prediction*, Trans. American Society of Mechanical Engineers, Series D, Journal of Basic Engineering, Vol. 83D, pp. 95-108.

Siegwart. R. & Nourbakhsh. I. R. (2004). *Introduction to Autonomous Mobile Robots,* MIT Press.

Howard. A. (2005). *Multi-robot Simultaneous Localization and Mapping Using Particle Filters,* Robotics: Science and Systems I, pp. 201-208.

Anderson. B. D. O. & Moore. B. J. (1979). *Optimal Filtering,* Englewood Cliffs: Prentice-Hall.

Gordon. N. J. ; Salmon. D. J. & Smith. A. F. M. (1993). *A Novel Approach For Nonlinear/non-Gaussian Bayesian State Estimation*, IEE Proceedings on Radar and Signal Processing, 140, 107-113.

Doucet. A; Freitas. N. de. & Gordon. N. J. (2001). *Sequential Monte Carlo Methods in Practice,* Springer- Verlag.

Ristic. B; Arulampalam. S. & Gordon. N. J. (2004). *Beyond the Kalman Filter*, Artech House.

Ng. W; Li. J; Godsill. S. & Vermaak. J. (2004). *A Hybrid Approach For Online Joint Detection And Tracking For Multiple Targets*, In the Proceedings of IEEE Aerospace Conference.

Gustafsson. F; Gunnarsson. F; Bergman. N; Forssell. U; Jansson. J; Karlsson. R. & Nordlund. P. J. (2002). *Particle Filters For Positioning, Navigation, and Tracking,* IEEE Transactions on Signal Processing, Vol. 50, No. 2.

Freitas. N. de. (1999). *Bayesian Methods For Training Neural Networks,* PhD Thesis, Trinity College, The University of Cambridge.

Andrieu. C; Doucet. A; Singh. S. S & Tadic. V. (2004). *Particle Methods For Change Detection, System Identification and Control,* Proceedings of IEEE, Vol. 92. No. 3.

Sarkka. S; Vehtari. A. & Lampininen. J. (2005). *Rao-Blackwellized Particle Filter for Multiple Target Tracking,* Information Fusion Journal, Vol. 8, Issue. 1, Pages 2-15.

Vermaak. J; Godsill. S. J. & Perez. P. (2005). *Monte Carlo Filtering for Multi- Target Tracking and Data Association*, IEEE Transactions on Aerospace and Electronic Systems, Vol. 41, No. 1, pp. 309–332.

Li. J; Ng. W. & Godsill. S. (2007). *Online Multiple Target Tracking and Sensor Registration Using Sequential Monte Carlo Methods*, In the Proceedings of IEEE Aerospace Conference.

Fortmann. T. E; Bar-Shalom. Y. & Scheffe. M. (1983). *Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association*, IEEE Journal of Oceanic Engineering, Vol. 8, pp. 173-184.

Schulz. D; Burgard. W; Fox. D. & Cremers. A. B. (2003). *People Tracking with a Mobile Robot Using Sample-based Joint Probabilistic Data Association Filters*, International Journal of Robotics Research (IJRR), 22(2).

Oh. S; Russell. S. & Sastry. S. (2004). *Markov Chain Monte Carlo Data Association for Multiple-Target Tracking*, In Proc. of the IEEE International Conference on Decision and Control (CDC), Paradise Island, Bahamas.

Singh. S. S; Kantas. N; Vo. B. N; Doucetc. A. & Evans. R. J. (2007). *Simulation-based Optimal Sensor Scheduling with Application to Observer Trajectory Planning*, Automatica, 43, 817 – 830.

Del Moral. P; Doucet. A & Jasra. A (2006). *Sequential Monte Carlo Samplers,* J. R. Statistics, Soc. B, Vol. 68, part 3, pp. 411-436.

Ikoma. N; Ichimura. N; Higuchi. T. & Maeda. H. (2001). *Particle Filter Based Method for Maneuvering Target Tracking*, IEEE International Workshop on Intelligent Signal Processing, Budapest, Hungary, pp.3-8,.

Pitre. R. R; Jilkov. V. P. & Li. X. R. (2005). *A comparative study of multiple-model algorithms for maneuvering target tracking*, Proc. 2005 SPIE Conf. Signal Processing, Sensor Fusion, and Target Recognition XIV, Orlando, FL.

Wang. X. L. (1997). *A Course in Fuzzy systems And Control,* Prentice-Hall, Inc, International
          Edition.

Tools in Artificial Intelligence

Edited by Paula Fritzsche

This book offers in 27 chapters a collection of all the technical aspects of specifying, developing, and evaluating the theoretical underpinnings and applied mechanisms of AI tools. Topics covered include neural networks, fuzzy controls, decision trees, rule-based systems, data mining, genetic algorithm and agent systems, among many others. The goal of this book is to show some potential applications and give a partial picture of the current state-of-the-art of AI. Also, it is useful to inspire some future research ideas by identifying potential research directions. It is dedicated to students, researchers and practitioners in this area or in related fields.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Aliakbar Gorji Daronkolaei, Vahid Nazari, Mohammad Bagher Menhaj, and Saeed Shiry (2008). A Joint Probability Data Association Filter Algorithm for Multiple Robot Tracking Problems, Tools in Artificial Intelligence, Paula Fritzsche (Ed.), ISBN: 978-953-7619-03-9, InTech, Available from: http://www.intechopen.com/books/tools_in_artificial_intelligence/a_joint_probability_data_association_filter_algorithm_for_multiple_robot_tracking_problems

INTECH
open science | open minds