Chapter 2

Foundations of Bayesian Multiple Target Tracking

The problem of estimating the position and the motion of a certain entity in the space is generally defined as tracking. This chapter introduces some of the most common techniques used for the tracking of multiple targets. First, some classic motion and observation models are illustrated in Section 2.1 an 2.2 respectively. Then, Section 2.3 introduces the basics of Bayesian estimation and the implementation of three popular filters, namely Extendend Kalman Filter, Unscented Kalman Filter and Sampling Importance Resampling Particle Filter. Finally, the data association problem is explained in Section 2.4, including some of the most famous algorithms to deal with it.

2.1 Motion Models

There are many ways to model the motion of a target in the space. A number of models have been proposed for tracking a target moving in the horizontal plane [6, 64]. This section illustrates 2D motion models representing the discrete time evolution of the target state vector. Those described next are some of the most popular solutions to predict the position of a target in the Cartesian space, in particular with regard to applications for people tracking.

2.1.1 Brownian Motion

One of the simplest models for target tracking is referred to as Brownian motion, or discrete Wiener process [8, 92]. The state vector consist on the 2D position $[x_k, y_k]^T$, where k is the current time step, and the relative model is described as follows:

$$\begin{cases} x_k &= x_{k-1} + n^x_{k-1} \\ y_k &= y_{k-1} + n^y_{k-1} \end{cases} \tag{2.1}$$

where the noises n^x_{k-1} and n^y_{k-1} are zero-mean Gaussians.

Such a model avoids complications that derive from considering velocity and orientation. At the same time, it is a good approximation of the human walking behaviour, as it is not influenced by sudden changes in direction. However, the accuracy of this model is strictly dependent on the frequency of the observations, which must be very high. Also, without the velocity estimation, this model cannot handle short occlusions, for example when a target moves behind an obstacle or another target.

2.1.2 Constant Velocity

Various (noiseless) models for tracking a target in a horizontal plane can be described by the following continuous curvilinear-motion model, which is derived from kinematics [64]:

$$\begin{cases} \dot{x}(t) &= v(t)\cos\phi(t) \\ \dot{y}(t) &= v(t)\sin\phi(t) \\ \dot{v}(t) &= a_t(t) \\ \dot{\phi}(t) &= \dfrac{a_n(t)}{v(t)} \end{cases} \tag{2.2}$$

where $x(t)$ and $y(t)$ are the Cartesian coordinates, $v(t)$ the velocity and $\phi(t)$ the orientation of the target at time t. The family of models derived from Equation (2.2) depends on the tangential and normal accelerations, $a_t(t)$ and $a_n(t)$, which can be null or not.

When $a_t(t)$ and $a_n(t)$ are null, Equation (2.2) gives origin to the so-called Constant Velocity (CV) model. Its discrete-time version can be written as follows [8]:

$$\begin{cases} x_k &= x_{k-1} + v_{k-1}\Delta t_k \cos\phi_{k-1} + n^x_{k-1} \\ y_k &= y_{k-1} + v_{k-1}\Delta t_k \sin\phi_{k-1} + n^y_{k-1} \\ v_k &= v_{k-1} + n^v_{k-1} \\ \phi_k &= \phi_{k-1} + n^\phi_{k-1} \end{cases} \tag{2.3}$$

where $\Delta t_k = t_k - t_{k-1}$ is the time interval, and the state vector is given by the 2D position (x_k, y_k), the velocity v_k and the orientation ϕ_k. The quantities n^x_{k-1}, n^y_{k-1}, n^v_{k-1} and n^ϕ_{k-1} are Gaussian noises. The CV model is most often written using a Cartesian representation of the velocity:

$$\begin{cases} x_k &= x_{k-1} + \dot{x}_{k-1}\Delta t_k + n^x_{k-1} \\ \dot{x}_k &= \dot{x}_{k-1} + n^{\dot{x}}_{k-1} \\ y_k &= y_{k-1} + \dot{y}_{k-1}\Delta t_k + n^y_{k-1} \\ \dot{y}_k &= \dot{y}_{k-1} + n^{\dot{y}}_{k-1} \end{cases} \tag{2.4}$$

where $\dot{x}_k$ and $\dot{y}_k$ are the velocity components in x and y, with their relative noises $n^{\dot{x}}_{k-1}$ and $n^{\dot{y}}_{k-1}$.

This model assumes that the target moves with a nearly-constant rectilinear velocity, which is obviously not true in general. Its simplicity, however, and the fact that it can deal with short occlusions (thanks to the non-null velocity), make the CV model a good compromise in many tracking applications.

2.1.3 Constant Acceleration and Constant Turn-rate Models

Two other models can be derived from Equation (2.2) when either $a_t(t)$ or $a_n(t)$ is not null, which are respectively the Constant Acceleration (CA) and the Constant Turn-rate (CT) models. The equations for these models are slightly more complicated and, since not of interest for the current research, they are omitted here for sake of brevity. It is sufficient to know that the CT model can be used to describe a curvilinear trajectory and that it has been successfully applied in multiple-models algorithms for aircraft tracking. The interested reader can find a detailed description in [64].

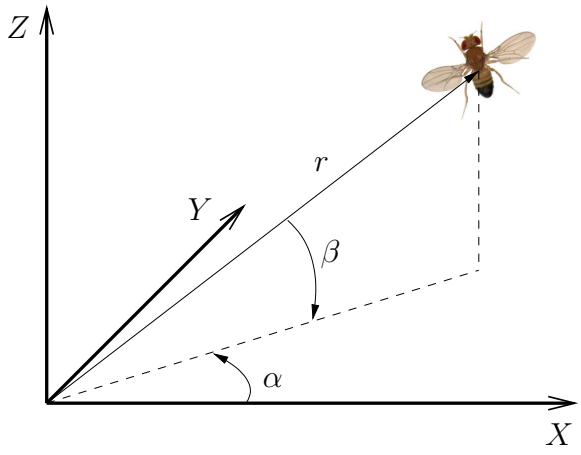

Figure 2.1: Range, bearing and elevation measurements.

2.2 Observation Models

The model to describe a particular measurement depends on the relative sensor. In many cases, the information provided by the devices used for target tracking belongs to one or more of the following classes: range, bearing and (angle of) elevation. The range is the distance measured between sensor and target, the bearing is the angle of the target's direction on the horizontal plane, and the elevation the angle in the vertical plane.

There are several conventions to represent these measurements. The equations described next are derived from the schematic representation in Figure 2.1. In particular, given the 3D position (x, y, z) of a generic target in the sensor's frame of reference, r is the range measurement, α the bearing and β the elevation.

The range measurement is the euclidean distance from the origin, calculated as follows:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{2.5}$$

The bearing is the angle, in the horizontal plane, obtained by the following equation[1]:

$$\alpha = \tan^{-1}\left(\frac{y}{x}\right) \tag{2.6}$$

Finally, the elevation angle is calculated from the z coordinate and the distance in the horizontal plane as follows:

$$\beta = -\tan^{-1}\left(\frac{z}{\sqrt{x^2 + y^2}}\right) \tag{2.7}$$

Each one of the observation models in Equations (2.5), (2.6) and (2.7) includes usually an additive noise term, here omitted for simplicity, that takes into account the measurement error.

2.3 Recursive Bayesian Estimation

The most popular methods for dynamic state estimation belong to the family of recursive Bayesian estimators, which include Kalman filters [53, 105] and sequential Monte Carlo estimators [2], also known as particle filters. These estimate the target position recursively, combining the expected state information with the current observations from the sensors.

[1]Note that, in this case, the function $\mathrm{atan2}$ should be used, thus to retrieve the angle in the proper quadrant.

In the discrete-time domain, for a general tracking application, the evolution of the target state can be described by the following general model:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) \tag{2.8}$$

where $\mathbf{x}_k$ is the state vector at the current time step k and $\mathbf{w}_{k-1}$ is white noise. The relative observations are generally described by another model with additive noise:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{2.9}$$

where $\mathbf{z}_k$ is the observation vector and $\mathbf{v}_k$ is white noise, mutually independent from $\mathbf{w}_{k-1}$. The function $\mathbf{f}$ and $\mathbf{h}$ can be non-linear.

If $\mathbf{Z}_k = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ is the set of observations up to time k, the prior probability density $p(\mathbf{x}_k|\mathbf{Z}_{k-1})$ can be expressed as follows:

$$p(\mathbf{x}_k|\mathbf{Z}_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Z}_{k-1})\, p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})\, d\mathbf{x}_{k-1} \tag{2.10}$$

$$= \int p(\mathbf{x}_k|\mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})\, d\mathbf{x}_{k-1} \tag{2.11}$$

where the transitional density $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Z}_{k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is determined by the Markovian prediction model in Equation (2.8). Then, applying Bayes' rule, the posterior density is given by the following equation:

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{Z}_k) &= p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{Z}_{k-1}) \\ &= \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{Z}_{k-1})\, p(\mathbf{x}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} \\ &= \frac{p(\mathbf{z}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} \end{aligned} \tag{2.12}$$

Note that, at the nominator, $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{Z}_{k-1}) = p(\mathbf{z}_k|\mathbf{x}_k)$ because $\mathbf{z}_k$ is completely described by the observation model in Equation (2.9), which depends only on the current state $\mathbf{x}_k$ and the noise $\mathbf{v}_k$. The denominator is just a normalization factor calculated as follows:

$$p(\mathbf{z}_k|\mathbf{Z}_{k-1}) = \int p(\mathbf{z}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{Z}_{k-1})\, d\mathbf{x}_k \tag{2.13}$$

Equations (2.10) and (2.12) are called, respectively, prediction and correction (or update) of the recursive Bayesian estimation. The desired estimate is usually obtained, at the end of every predict-correct iteration, by the *Minimum Mean-Square Error* (MMSE) value, i.e. the conditional mean $\hat{\mathbf{x}}_k \triangleq \mathbb{E}[\mathbf{x}_k|\mathbf{Z}_k]$.

2.3.1 Extended Kalman Filter

The Kalman filter was initially proposed in [57] and, although originally not formulated as such, it has been later shown to belong to the more general class of Bayesian estimators [9]. It was also proved to be optimal in case of linear systems with Gaussian noises. In case of non-linearities, the Extended Kalman Filter (EKF) provides an approximate solution, applying the same equations to linearized system models. This can give good results if the linearization is sufficiently accurate to describe the system, but fails badly if it is not.

Given $\mathbf{x}_k = [x_1, \ldots, x_n]_k^T$ and $\mathbf{z}_k = [z_1, \ldots, z_m]_k^T$, the prediction and observation models in Equations (2.8) and (2.9) can be rewritten as follows:

$$\mathbf{x}_k = \begin{bmatrix} f_1(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) \\ \vdots \\ f_n(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) \end{bmatrix} \tag{2.14}$$

$$\mathbf{z}_k = \begin{bmatrix} h_1(\mathbf{x}_k) \\ \vdots \\ h_m(\mathbf{x}_k) \end{bmatrix}_k + \mathbf{v}_k \tag{2.15}$$

At each time step, the Jacobians $\mathbf{F}_k$ and $\mathbf{G}_k$ of the prediction model are calculated, where the elements are given by the partial derivatives about the previous estimate [105]:

$$\mathbf{F}_k^{(i,j)} = \frac{\partial f_i}{\partial x_j}(\hat{\mathbf{x}}_{k-1}, 0) \tag{2.16}$$

$$\mathbf{G}_k^{(i,j)} = \frac{\partial f_i}{\partial w_j}(\hat{\mathbf{x}}_{k-1}, 0) \tag{2.17}$$

The prediction stage consists in calculating the *a-priori* estimate $\hat{\mathbf{x}}_k^-$ and the covariance matrix $\mathbf{P}_k^-$ of its error:

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, 0) \tag{2.18}$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_{k-1} \mathbf{G}_k^T \tag{2.19}$$

where the matrix $\mathbf{Q}_{k-1}$ is the covariance of the state noise $\mathbf{w}_{k-1}$. In case of additive noise with constant covariance, the product $\mathbf{G}_k \mathbf{Q}_{k-1} \mathbf{G}_k^T$ is just a constant matrix $\mathbf{Q}$.

Then, the Jacobian $\mathbf{H}_k$ of the observation model has to be calculated as follows:

$$\mathbf{H}_k^{(l,j)} = \frac{\partial h_l}{\partial x_j}(\hat{\mathbf{x}}_k^-) \tag{2.20}$$

The correction part computes the Kalman gain $\mathbf{K}_k$ and finally calculates *a-posteriori* estimate $\hat{\mathbf{x}}_k$ and a relative error covariance $\mathbf{P}_k$ as follows:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \tag{2.21}$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{2.22}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k \right) \tag{2.23}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^T \tag{2.24}$$

where the term $(\mathbf{z}_k - \hat{\mathbf{z}}_k)$, with $\hat{\mathbf{z}}_k = \mathbf{h}(\hat{\mathbf{x}}_k^-)$, is the difference between real and predicted measurements, also called *innovation*. The quantity $\mathbf{S}_k$ in Equation (2.21) is the innovation covariance and $\mathbf{R}_k$ is the covariance matrix of the observation noise $\mathbf{v}_k$.

2.3.2 Unscented Kalman Filter

To overcome the problem of the linearization, which could introduce large errors and require the computation of big Jacobian matrices, the Unscented Kalman Filter (UKF) makes use of another approximation, called Unscented Transformation (UT). This is based on the idea that it is generally easier and more accurate to approximate probability distributions than non-linear functions [54]. The UT captures mean and covariance of a probability distribution with carefully chosen weighted points, called *sigma points*. These differ from the points of particles filters in that they are not randomly sampled and do not have to lie in the interval $[0, 1]$.

From the state $\mathbf{x}$ of size n, and its error covariance $\mathbf{P}$, the $2n + 1$ sigma points $\boldsymbol{\mathcal{X}}_i$ and associated weights W_i of the UT are calculated using the following equations [53]:

$$\begin{aligned} \boldsymbol{\mathcal{X}}_0 &= \mathbf{x} & W_0 &= \rho/(n+\rho) \\ \boldsymbol{\mathcal{X}}_i &= \mathbf{x} + \left[\sqrt{(n+\rho)\,\mathbf{P}}\right]_i & W_i &= [2\,(n+\rho)]^{-1} \\ \boldsymbol{\mathcal{X}}_{i+n} &= \mathbf{x} - \left[\sqrt{(n+\rho)\,\mathbf{P}}\right]_i & W_{i+n} &= [2\,(n+\rho)]^{-1} \end{aligned} \tag{2.25}$$

where $i = 1, \ldots, n$. The term $\left[\sqrt{(n+\rho)\mathbf{P}}\right]_i$ is the i^{th} column or row of the matrix square root of $\mathbf{P}$, and ρ is a parameter for tuning the higher order moments of the approximation ($n + \rho = 3$ for Gaussian distributions).

Using these points, mean and covariance of a generic (non-linear) transformation $\mathbf{y} = \mathbf{g}(\mathbf{x})$ are calculated as follows:

$$\boldsymbol{\mathcal{Y}}_i = \mathbf{g}\left(\boldsymbol{\mathcal{X}}_i\right) \tag{2.26}$$

$$\mathbf{y} = \sum_{i=0}^{2n} W_i \boldsymbol{\mathcal{Y}}_i \tag{2.27}$$

$$\mathbf{P}_{\mathbf{yy}} = \sum_{i=0}^{2n} W_i \left[\boldsymbol{\mathcal{Y}}_i - \mathbf{y}\right]\left[\boldsymbol{\mathcal{Y}}_i - \mathbf{y}\right]^T \tag{2.28}$$

As shown in [54], this procedure yields to a projected mean and covariance that are correct up to the second order, thus better than the linearization used by the EKF, yet keeping the same computational complexity.

In the most general form of the UKF, the state is augmented to include prediction and observation noises. The estimation process starts with the generation of the sigma points $\boldsymbol{\mathcal{X}}_{i_{k-1}}$, applying Equation (2.25) to the previous estimate $\hat{\mathbf{x}}_{k-1}$. The a-priori mean $\hat{\mathbf{x}}_k^-$ and covariance $\mathbf{P}_k^-$ are then predicted as follows:

$$\hat{\mathbf{x}}_{k-1} \xrightarrow{\text{UT}} \left\{\boldsymbol{\mathcal{X}}_{i_{k-1}}\right\}_{i=0}^{2n} \tag{2.29}$$

$$\boldsymbol{\mathcal{X}}_{i_k}^- = \mathbf{f}(\boldsymbol{\mathcal{X}}_{i_{k-1}}) \quad \text{for } i = 0, \ldots, 2n \tag{2.30}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2n} W_i \boldsymbol{\mathcal{X}}_{i_k}^- \tag{2.31}$$

$$\mathbf{P}_k^- = \sum_{i=0}^{2n} W_i \left[\boldsymbol{\mathcal{X}}_{i_k}^- - \hat{\mathbf{x}}_k^-\right]\left[\boldsymbol{\mathcal{X}}_{i_k}^- - \hat{\mathbf{x}}_k^-\right]^T \tag{2.32}$$

The expected measurement is calculated applying the observation model to the new sigma points in Equation (2.30):

$$\boldsymbol{\mathcal{Z}}_{i_k} = \mathbf{h}(\boldsymbol{\mathcal{X}}^-_{i_k}) \quad \text{for } i = 0, \dots, 2n \tag{2.33}$$

$$\hat{\mathbf{z}}_k = \sum_{i=0}^{2n} W_i \boldsymbol{\mathcal{Z}}_{i_k} \tag{2.34}$$

The innovation covariance $\mathbf{S}_k$ and the cross-correlation $\mathbf{C}_k$ are then computed, followed by the gain $\mathbf{K}_k$:

$$\mathbf{S}_k = \sum_{i=0}^{2n} W_i \left[\boldsymbol{\mathcal{Z}}_{i_k} - \hat{\mathbf{z}}_k\right] \left[\boldsymbol{\mathcal{Z}}_{i_k} - \hat{\mathbf{z}}_k\right]^T \tag{2.35}$$

$$\mathbf{C}_k = \sum_{i=0}^{2n} W_i \left[\boldsymbol{\mathcal{X}}^-_{i_k} - \hat{\mathbf{x}}^-_k\right] \left[\boldsymbol{\mathcal{Z}}_{i_k} - \hat{\mathbf{z}}_k\right]^T \tag{2.36}$$

$$\mathbf{K}_k = \mathbf{C}_k \mathbf{S}_k^{-1} \tag{2.37}$$

Finally, the a-posteriori estimate $\hat{\mathbf{x}}_k$ and relative covariance $\mathbf{P}_k$ are determined applying the same Equations (2.23) and (2.24) previously used for the EKF.

2.3.3 SIR Particle Filter

Particle filters are recursive Bayesian estimators that make use of Monte Carlo methods to approximate and transform probability distributions [2, 33, 81]. The major advantages of such filters are their independence from the non-linearities of a system and capability to approximate any kind of probability distribution, including multimodal cases. The drawback is that a large number of particles is normally required for a good estimation, with a consequent negative effect on the computational cost.

In particle filters, the posterior of the state, introduced in Equation (2.12), is approximated by the weighted sum of N samples $\mathbf{x}^i_k$:

$$p(\mathbf{x}_k|\mathbf{Z}_k) \approx \sum_{i=1}^{N} w^i_k \delta(\mathbf{x}_k - \mathbf{x}^i_k) \tag{2.38}$$

where $\delta(\cdot)$ is the Dirac delta measure. The samples $\mathbf{x}^i_k$ are drawn from a known *importance density* $q(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k)$, and their weights are calculated recursively as follows:

$$w^i_k \propto w^i_{k-1} \frac{p(\mathbf{z}_k|\mathbf{x}^i_k) p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1})}{q(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k)} \tag{2.39}$$

It can be proved that for $N \to \infty$ the approximation in Equation (2.38) tends to the true posterior $p(\mathbf{x}_k|\mathbf{Z}_k)$.

There are many different implementations of particle filters, however the Sampling Importance Resampling (SIR) algorithm is probably the most popular, due to its simplicity. This estimator, originally proposed in [46] with the name of "bootstrap" filter, makes use of the transitional prior as importance density:

$$q(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}, \mathbf{z}_k) = p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1}) \tag{2.40}$$

The importance density can be determined therefore from the state model in Equation (2.8). This choice simplifies also the calculus of the weights, which are given by the current measurement likelihood:

$$w_k^i \propto w_{k-1}^i \, p(\mathbf{z}_k|\mathbf{x}_k^i) \tag{2.41}$$

where $p(\mathbf{z}_k|\mathbf{x}_k^i)$ depends on the observation model of Equation (2.9). At the end of each iteration, the SIR algorithm performs a resample step that eliminates all the particles with very small weights and, from the remaining ones, generates new samples equally weighted.

Like for the previous ones, the SIR estimation has an iterative predict-correct sequence. The prediction part generates new particles, from the previous ones, using Equation (2.8) and samples drawn from the probability distribution of the state noise. Then, as soon as a new measurement is available, the correction is performed calculating the weights in Equation (2.41) and using them to obtain the approximated posterior with Equation (2.38). The particles are finally resampled for the next iteration. A detailed explanation of SIR and other particle filters is given in [2, 81].

2.4 Data Association

When the entities to track are more than one, it is necessary to determine which target the current measurements belong to. These will be used then to update the relative estimators, which are usually as many as the targets being tracked. The process is called *data association*, and plays a critical role in every multitarget tracking system. Very often, indeed, a single error in the measurement assignment could irreparably compromise the tracking of one or more targets. The following sections review some of the most popular algorithms for gating and data association that in many cases serve also as basis for more advanced and application-tailored solutions.

2.4.1 Validation Gating

Before the actual assignment, the sensor measurements are initially passed through a *gating* process in order to reduce the number of false positives and the computational complexity of the data association algorithm. The process includes the definition of a *validation region*, built around the expected observation of the considered target, where new measurements could possibly lie on. This region could be simply determined, for example, taking the distance covered, in the considered time interval, by a moving target at its maximum speed.

A more general approach for determining the validation region of an observation takes into account the uncertainty of the estimate and of the sensor measurement. In case of measurements normally (Gaussian) distributed, an opportune region can be defined, around the expected observation $\hat{\mathbf{z}}_j$, such that the following relation holds [7]:

$$(\mathbf{z}_i - \hat{\mathbf{z}}_j)^T \, \mathbf{S}_{ij}^{-1} \, (\mathbf{z}_i - \hat{\mathbf{z}}_j) \leqslant \lambda \tag{2.42}$$

where λ is a threshold, $\mathbf{z}_i$ is a real measurement and $\mathbf{S}_{ij}$ is the covariance matrix of the difference $(\mathbf{z}_i - \hat{\mathbf{z}}_j)$. Under the given assumptions, the left term in Equation (2.42), which is also known as *Mahalanobis distance*, has a χ^2 (chi-square) distribution. Therefore, it is possible to determine the value of λ, from tables of this distribution, simply knowing the "degrees of freedom" (i.e. dimension of the observation vector) and setting a probability for the measurement in the validation region to be actually generated by the relative target. Note that the same gating approach is usually adopted even in case the assumption of Gaussianity does not hold.

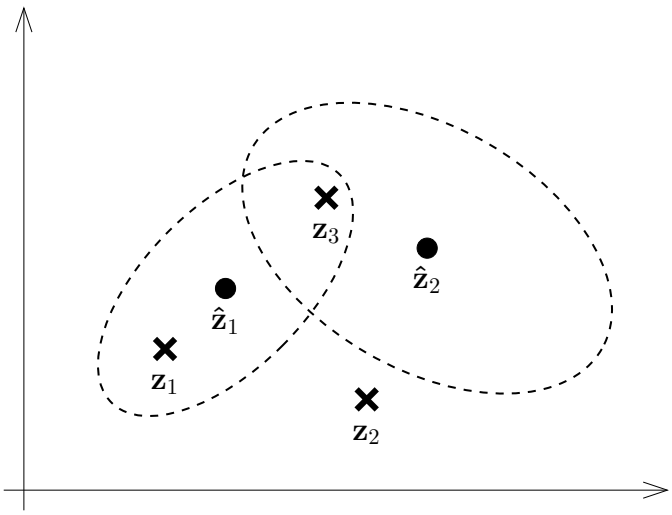

Figure 2.2: Example of gating procedure.

Of course, the gating procedure alone is not sufficient to guarantee the correct assignment of new measurements, since in many real situations the validation regions of different targets intersect with each other, generating ambiguities. The concept can be simply explained from the example in Figure 2.2, where the circles are the predicted observations with their relative (elliptical) validation regions, and the crosses are the new measurements from the sensor that need to be associated. Suppose to have two tracks, for which the expected observations are $\hat{\mathbf{z}}_\mathbf{1}$ and $\hat{\mathbf{z}}_\mathbf{2}$. For each one of these, a validation region is defined to mark the area where a possible real measurement could lie. Then, if the current sensor measurements are $\mathbf{z}_1$, $\mathbf{z}_2$ and $\mathbf{z}_3$, three different situations may occur: (a) the measurement $\mathbf{z}_1$ lies only on the validation region of $\hat{\mathbf{z}}_\mathbf{1}$; (b) the measurement $\mathbf{z}_2$ is not generated by any target; (c) $\mathbf{z}_3$ can belong either to one track or the other (sometimes to both). To resolve the latter assignment, an appropriate data association algorithm has to be applied.

2.4.2 Nearest Neighbour

The Nearest Neighbour (NN) data association [7] is probably the simplest and most straightforward algorithm. As the name suggests, every association is made assigning to each target the measurement $\mathbf{z}_i$ that is closest to its expected observation $\hat{\mathbf{z}}_j$. The distance between these two vectors is usually a statistical measure that takes into account the uncertainty of the track estimate and the sensor reading. For example, the Mahalanobis distance defined in Equation (2.42) is a popular choice, since it is already available from the previous gating process. However, such a distance might not be appropriate in case of large uncertainties. Indeed, an assignment with difference $(\mathbf{z}_1 - \hat{\mathbf{z}}_j)$ and a large covariance $\mathbf{S}_{1j}$ will always be preferred to another $(\mathbf{z}_2 - \hat{\mathbf{z}}_j)$ with a small $\mathbf{S}_{2j}$, even if $||\mathbf{z}_1 - \hat{\mathbf{z}}_j|| \gg ||\mathbf{z}_2 - \hat{\mathbf{z}}_j||$ would suggest that the second choice is better. The problem can be solved using another measure of similarity, defined as follows [100]:

$$s_{ij} = \frac{1}{(2\pi)^{n/2}\,|\mathbf{S}_{ij}|^{1/2}}\ \exp\left[-\frac{1}{2}\left(\mathbf{z}_i - \hat{\mathbf{z}}_j\right)^T \mathbf{S}_{ij}^{-1}\left(\mathbf{z}_i - \hat{\mathbf{z}}_j\right)\right] \tag{2.43}$$

where n is the dimension of the observation vector. The assignments will be done to maximize the similarity measure in Equation (2.43), which balances the difference between real and predicted observations with its relative uncertainty.

In practice, at every time step, the data association procedure starts with the construction of a *validation matrix* $\mathbf{\Omega}$ of binary elements ω_{ij}, where $i = 1, \ldots, M$ is the index of the real measurement and $j = 1, \ldots, N$ is that one of the predicted observation (i.e. target's number). For example, with regard to Figure 2.2, the relative validation matrix could be written as follows:

$$\mathbf{\Omega} = \begin{array}{c} \\ \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{array} \begin{array}{c} \begin{array}{cc} t_1 & t_2 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \end{array} \tag{2.44}$$

where $\omega_{ij} = 1$ means that the measurement $\mathbf{z}_i$ lies inside the validation region of target t_j. For each one of these elements, a similarity measure s_{ij} can be calculated using Equation (2.43). The actual data association starts therefore with the assignment $\langle \mathbf{z}_{i^*},\ t_{j^*} \rangle$ for which this similarity is maximum. The relative indexes i^* and j^* are removed from further consideration, and the association continues with the remaining elements, until there are no more measurements or targets available. So, in the example above, if the similarity measures are $s_{11} > s_{31} > s_{32}$, the algorithm is completed in two steps:

$$\begin{bmatrix} \boxed{s_{11}} & - \\ - & - \\ s_{31} & s_{32} \end{bmatrix} \quad \text{remove row } i = 1 \text{ and column } j = 1 \tag{2.45}$$

$$\begin{bmatrix} - & - \\ - & - \\ - & \boxed{s_{32}} \end{bmatrix} \quad \text{remove row } i = 3 \text{ and column } j = 2 \tag{2.46}$$

and the resulting assignments are $\langle \mathbf{z}_1,\ t_1 \rangle$ and $\langle \mathbf{z}_3,\ t_2 \rangle$.

NN is therefore a one-to-one association, where a target can have only one measurement, and a measurement can be generated only by one target[2]. At the end of the procedure, the measurements which have not been associated to any target (e.g. $\mathbf{z}_2$ in the previous example) are normally used for the creation of new tracks.

2.4.3 Joint Probabilistic Data Association

In the Joint Probabilistic Data Association (JPDA) [7, 39], all the measurements are considered as possibly generated by any of the targets, or by none of them. The validation matrix, in this case, includes an additional column t_0 to indicate that the measurement does not belong to any of the current targets. So, with regard to Figure 2.2, the matrix would be the following:

$$\mathbf{\Omega} = \begin{array}{c} \\ \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_0 & t_1 & t_2 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \end{array} \tag{2.47}$$

First, it is necessary to generate all the possible *feasible association* matrices $\hat{\mathbf{\Omega}}_h$, calculated from $\mathbf{\Omega}$ following the next rules:

[2]To distinguish from the case where one measurement can be assigned to several targets, this algorithm is also referred to as Global Nearest Neighbour (GNN).

- only one unit per row (i.e. one target per measurement)
- no more than one unit per column (i.e. no more than one measurement per target), with the exception of column t_0 that can have several measurements.

Therefore, in the example, the feasible associations are described by the following five matrices:

$$\hat{\mathbf{\Omega}}_1=\begin{bmatrix}1&0&0\\1&0&0\\1&0&0\end{bmatrix}\ \hat{\mathbf{\Omega}}_2=\begin{bmatrix}0&1&0\\1&0&0\\1&0&0\end{bmatrix}\ \hat{\mathbf{\Omega}}_3=\begin{bmatrix}1&0&0\\1&0&0\\0&1&0\end{bmatrix}\ \hat{\mathbf{\Omega}}_4=\begin{bmatrix}1&0&0\\1&0&0\\0&0&1\end{bmatrix}\ \hat{\mathbf{\Omega}}_5=\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix} \tag{2.48}$$

Then, the probability of each feasible association is calculated using the following formula [39]:

$$P_{\hat{\mathbf{\Omega}}_h}=\frac{C^\phi}{\Gamma}\prod_{\mathcal{A}_h}\frac{\exp\left[-\frac{1}{2}\left(\mathbf{z}_i-\hat{\mathbf{z}}_j\right)^T\mathbf{S}_{ij}^{-1}\left(\mathbf{z}_i-\hat{\mathbf{z}}_j\right)\right]}{(2\pi)^{n/2}\left|\mathbf{S}_{ij}\right|^{1/2}}\prod_{\delta_h}P_{Dj}\prod_{\bar{\delta}_h}(1-P_{Dj}) \tag{2.49}$$

where C and ϕ are respectively the density and the total number of false measurements, Γ is a normalization factor, $\mathcal{A}_h$ is the set of actual associations in $\mathbf{\Omega}_h$ (without t_0), P_{Dj} is the probability of detecting target t_j, δ_h is the set of detected targets and $\bar{\delta}_h$ the remaining ones.

Finally, the *association probabilities* are calculated, for every combination of measurement $\mathbf{z}_i$ and target t_j, summing over all the feasible associations as follows:

$$\begin{aligned}\beta_i^j&=\sum_h P_{\hat{\mathbf{\Omega}}_h}\hat{\omega}_{h,ij} && i=1,\dots,M \quad j=0,\dots,N\\ \beta_0^j&=1-\sum_{i=1}^{M}\beta_i^j && j=0,\dots,N\end{aligned} \tag{2.50}$$

where $\hat{\omega}_{h,ij}$ is a binary element of $\hat{\mathbf{\Omega}}_h$ and β_0^j is the probability that target t_j has no measurements. M and N are, respectively, the number of measurements and targets. For example, using the matrices in Equation (2.48), the association probability of $\langle \mathbf{z}_1,\ t_1\rangle$ would be calculated as follows:

$$\beta_1^1=P_{\hat{\mathbf{\Omega}}_2}+P_{\hat{\mathbf{\Omega}}_5} \tag{2.51}$$

In the standard JPDA filter [39], the association probabilities in Equation (2.49) are eventually used to calculate a weighted innovation and update the estimate with modified Kalman equations. Differently from the NN data association, therefore, this method uses a many-to-one/one-to-many approach, where several measurements can be generated by the same target, and several targets can give origin to the same measurement.

2.4.4 Multiple Hypothesis Tracking

The Multiple Hypothesis Tracking (MHT) [17, 80] is known to be, at least theoretically, one of the best data association algorithms, as it considers all the possible sequences of assignments performed over time. One of the most appealing feature of MHT is the automatic creation and removal of tracks, which in NN and JPDA, instead, needs to be solved apart by additional logic.

Suppose that h is the current hypothesis, constituted by a set of assignment at time k, and p is the *parent hypothesis* at time $k-1$. In the MHT, the set of all possible assignments at time

k is represented by an *ambiguity matrix* $\mathbf{\Omega}^k$, which depends on the previous hypothesis and on the current measurements. This $\mathbf{\Omega}^k$ can be constructed like the validation matrix in Section 2.4.2, adding a column t_0 for measurements not belonging to any of the current targets, and another column t_* to handle the case of measurements generated by new targets. With regard to the example in Figure 2.2, the ambiguity matrix would be as follows:

$$\mathbf{\Omega}^k = \begin{array}{c} \\ \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_0 & t_1 & t_2 & t_* \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array} \tag{2.52}$$

As for JPDA, it is possible to generate from $\mathbf{\Omega}^k$ a set of feasible associations $\hat{\mathbf{\Omega}}_h^k$ that have at most one unit per column, except for t_0 and t_*, and one unit per row[3]. Under some assumptions on the distribution of the involved quantities, the probability of each hypothesis can be calculated extending the previous $P_{\hat{\mathbf{\Omega}}_h}$ of the JPDA, in Equation (2.49), with the following expression:

$$P_{\hat{\mathbf{\Omega}}_h^k} = P_{\hat{\mathbf{\Omega}}_h} \left[\frac{G^{\varphi}}{\Gamma'} \prod_{\rho_h} P_{Rj} \prod_{\bar{\rho}_h} (1 - P_{Rj}) \right] P_{\hat{\mathbf{\Omega}}_p^{k-1}} \tag{2.53}$$

where G and φ are respectively the density and the total number of measurements generated by new targets, Γ' is a new normalization factor, P_{Rj} is the probability of removing target t_j, $P_{\hat{\mathbf{\Omega}}_p^{k-1}}$ is the probability of the parent hypothesis, ρ_h is the set of removed targets and $\bar{\rho}_h$ the remaining ones.

The evaluation of all possible assignments for all the existing parent hypotheses is computationally intractable. Therefore, the number of hypotheses generated has to be limited by clustering tracks with common measurements, pruning low probability hypotheses and merging the similar ones.

2.5 Summary

This chapter gave a panoramic introduction of the multitarget tracking problem, briefly explaining some common solutions based on probabilistic approaches. Some motion models for 2D tracking have been presented, followed by some observation models for range and directional sensors. Three classic Bayesian estimators have also been illustrated, namely EKF, UKF and SIR particle filter, highlighting advantages and disadvantages of each technique. Finally, a general explanation of the data association has been given, illustrating the gating procedure and three popular algorithms, NN, JPDA and MHT, to solve the problem.

Although this chapter covered, in a simplified manner, just a few of the many techniques available for multiple target tracking, the arguments treated provide the bases to understand the proposed solutions for people tracking, which are described in the following chapters.

[3] If the first and last columns of $\mathbf{\Omega}^k$ are replicated as many times as the number of measurements (i.e. M times), then there is only one single unit for each row or column, and searching for the best association can be thought as a problem of weighted bipartite graph matching [29].