Perception – I
Estimation and Fusion

Robotics Principia – GdR Robotics Winter School –

R. Chapuis

January 19, 2019

Contents

Chapter 1

Introduction

These lectures notes are prepared to provide minimal fundamentals to address estimation and fusion applications in robotics. We tried to present these basis using the data fusion as a main frame. These notes are organized as follows:

- We first introduce the Bayesian Estimation as the main fundamental angle stone to deal with minimal estimation issues,
- We then develop Dynamic Estimation in order to cope with real estimation issues robots have to face usually. In the section we'll talk about, among others, Kalman filtering.
- The fusion itself is then introduced. Notice, because of the lack of time / place, we don't address here important issues such as the **decision** or the **tracking** topics. In particular we'll address the centralized and un-centralized fusion and the data un-synchronization issues.
- Since nowadays, computer vision address very complex fusion issues, in particular the **Visual SLAM**, we just focus on the graphical models as tools to deal with complex systems such as SLAM's. In this part we'll show briefly Bayes Networks and the Factors Graphs.
- Finally A short Appendix gives few minimum reminders required to well understand these notes.

Chapter 2

Bayesian Estimation

2.1 Introduction

For a robot, embedding sensors is tremendous, but how can it make the best use of these (potentially numerous) incoming data ?

The answer to this question really depends on the main goal of the robot: does it need to grasp an object, detect and avoid obstacles, estimate its pose in the world ?

In this part we mainly concentrate on the **data fusion** aiming at giving **an estimation** of an unknown vector state x thanks to the sensors data regardless the real application. Hence, the decision step of the fusion won't be addressed here.

This first chapter will introduce the Bayesian estimation theory which is the basis to start with data fusion. We'll see afterwards what are estimators and the most commonly used.

2.2 Bayesian estimation principles

2.2.1 Introduction

Bayesian estimation is the basis for parametric data fusion. This section explains the main principles. Our problem is the following : we are looking for an estimation $\hat{x}$ of an unknown parameter x having a measurement z that is linked to x by eq. (2.1):

$$z = h(x, v) \tag{2.1}$$

Notice x is usually a vector as well as z. However we keep this notation for sake of simplicity.

At the moment, we only concentrate on the estimation of x at a given time regardless the eventual dynamic evolution of the robot (we'll address that question in §3).

2.2.2 Example

We wish to estimate parameters f_0, ϕ and A of a given sinusoidal noised signal (see figure 2.1) using the first z_n ($z \in [0, N[$) measurements of this signal:

$$z_n = A\sin(2\pi f_0 n + \phi) + v_n \qquad \text{with} \qquad n \in [0, N[$$

We can write:

$$z = h(x, v)$$

With: $z = (z_0, z_1, \cdots, z_{N-1})^\top$, $x = (f_0, \phi, A)^\top$ and $v = (v_0, v_1, \cdots, v_{N-1})^\top$.

We look for an **estimation** $\hat{x} = (\hat{f_0}, \hat{\phi}, \hat{A})^\top$ of $x = (f_0, \phi, A)^\top$ using $z = (z_0, z_1, \cdots, z_{N-1})^\top$. The estimation problem can be formulated as follows: **what is the best estimation** $\hat{x}$ **of** $x = (f_0, \phi, A)^\top$ **having** z **and knowing the statistical properties of the noise** v ?

Because of this noise, we'll need to use probability tools to solve this problem.

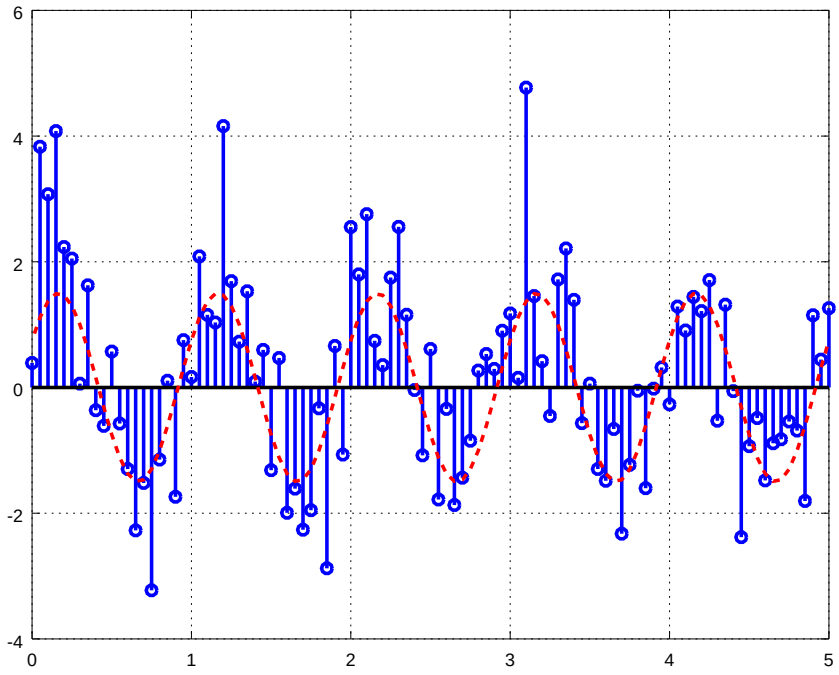


Figure 2.1: We seek for f_0, ϕ and A using noised data z_n (in blue). The optimal signal (unfortunately unknown) is represented by the red curve.

2.2.3 Bayes rules

We only consider the *continuous variables Bayes Rule* here. Consider x and y random values, we'll use their *pdf* [1]. We write:

- $p(x)$ and $p(y)$: *pdf* of x and y,
- $p(x, y)$: joint *pdf* of x and y,
- $p(x|y)$: conditional *pdf* of x having y

We have:

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x)$$

and:

$$\boxed{p(x|y) = \frac{p(x, y)}{p(y)} = \frac{p(y|x)p(x)}{p(y)}} \tag{2.2}$$

Since $p(y) = \int_{-\infty}^{+\infty} p(y|x)p(x)dx$, we'll have:

$$p(x|y) = \frac{p(y|x)p(x)}{\int_{-\infty}^{+\infty} p(y|x)p(x)dx}$$

Equation (2.2) can be generalized to multiple variables x_i for $i \in [1, n]$ and y_j for $j \in [1, m]$:

$$\boxed{p(x_1, \cdots, x_n|y_1, \cdots, y_m) = \frac{p(x_1, \cdots, x_n, y_1, \cdots, y_m)}{p(y_1, \cdots, y_m)} = \frac{p(y_1, \cdots y_m|x_1, \cdots, x_n)p(x_1, \cdots, x_n)}{p(y_1, \cdots, y_m)}} \tag{2.3}$$

[1] Probability Density Function

2.3 Likelihood function

2.3.1 Introduction

Actually we are looking for x having z measurement. So we wish to know $p(x|z)$. Considering equation (2.2), it is straightforward to obtain $p(x|z)$ using equation (2.4) :

$$\boxed{p(x|z) = \frac{p(z|x)p(x)}{p(z)}} \tag{2.4}$$

This is exactly what we want: getting x having z (notice that x and z are most of the time vectors). Even if getting an estimations $\hat{x}$ of x from $p(x|z)$ can be somewhat tricky (see §2.4) let's first concentrate on the components of equation (2.4). We have three parts: $p(z|x)$, $p(x)$ and $p(y)$.

- $p(z|x)$ is the link between the measurement z and the unknown x, this term is named **prior likelihood**,
- $p(x)$ is **prior knowledge** we have on x,
- $p(z)$ is the knowledge we have on z whatever x. Since $p(z)$ is no linked to x, $p(z)$ is rarely used in practice.

2.3.2 Likelihood function

$p(z|x)$ in equation (2.4) can represent two things:

1. the *pdf* of measurement z knowing x. Hence here z is a random variable while x is known, it is actually **the measurement characterization** knowing x. We can here make the link with eq. (2.1) we have already seen in §2.1. The noise v *pdf* combined with eq. (2.1) will provide $p(z|x)$.
2. $p(z|x)$ can also represent x while we got z: in this case $p(z|x)$ is the **likelihood function** of x. Here x is the random value and z is known.

   In order to mention explicitly that x is the random value, we write the likelihood function as:

   $$p(x; z|x)$$

   Some authors [11] even write:

   $$l(x; z) \triangleq p(x; z|x)$$

Let's see two examples to explain the differences between $p(z|x)$ and $p(x; z|x)$.

Example 1

Suppose the room temperature is $x = \theta = 20°$ and the sensor providing the measurement z is noised with a $\pm 1°$ uniform error. We can write eq. (2.1) as follows:

$$\theta = x = h(z, v) = z + v$$

with v the noise measurement whose *pdf* is given in figure 2.2-a-. Since $z = \theta - v$, we can draw the *pdf* $p(z|x)$ of z as shown in figure 2.2-b-.

Actually $x = \theta$ is unknown but we know the measurement z (even if it is noisy). Therefore we draw the **likelihood function** $p(\theta; z|\theta)$ as shown in figure 2.2-c-.

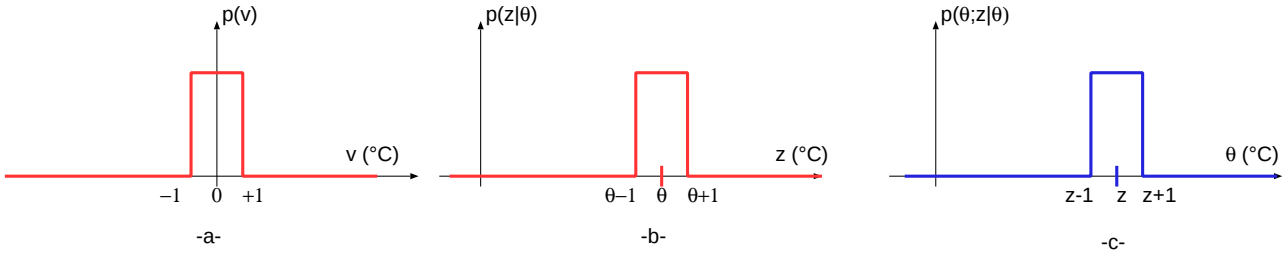


Figure 2.2: -a-: noise v *pdf* ; -b-: z measurement *pdf* knowing the real temperature θ: $p(z|\theta)$; -c-: likelihood Function of x having measurement z: $p(\theta; z|\theta)$

Example 2

Consider now a localization problem. We wish to know a vehicle pose $X = (x, y)^\top$ thanks to a GPS measurement $z = (x_{gps}, y_{gps})^\top$.

Figure 2.3 (left) shows the *pdf* $p(z|X)$ of measurement $z = (x_{gps}, y_{gps})^\top$ knowing the real position $X = (x, y)^\top$ and figure 2.3 (right) shows the *likelihood function* $p(X; z|X)$ of $X = (x, y)^\top$ having measurement $z = (x_{gps}, y_{gps})^\top$.

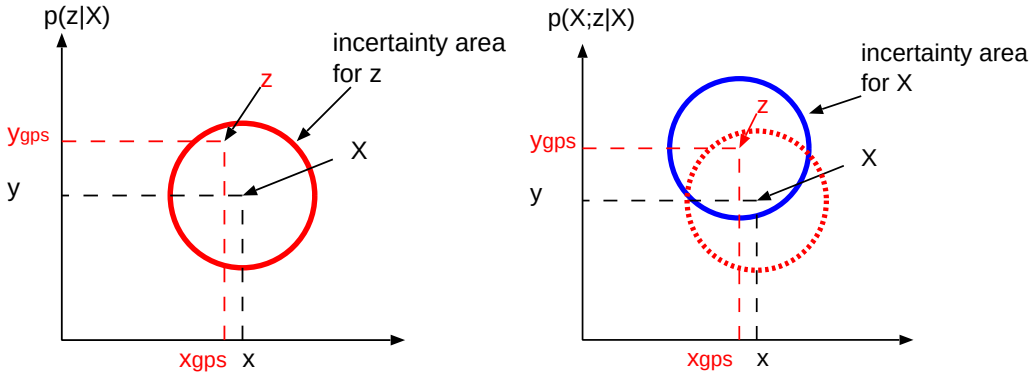


Figure 2.3: Left: $z = (x_{gps}, y_{gps})^\top$ *pdf* measurement knowing the real position $X = (x, y)^\top$. Right: *likelihood function* $p(X; z|X)$ of $X = (x, y)^\top$ having a measurement $z = (x_{gps}, y_{gps})^\top$

Subsidiary remarks

- since the random value in $p(x; z|x)$ is no longer z but x, then $p(x; z|x)$ is **no longer a *pdf*** . Hence :

$$\int p(z|x)dz = 1 \qquad \text{but} \qquad \int p(x; z|x)dx \neq 1$$

- Actually, $p(z|x)$ is necessary to characterize the sensor having a given value of x but the estimation will require $p(x; z|x)$.

- Now the Bayesian Estimation given in equation (2.4) can be rewritten as equation (2.5):

$$\boxed{p(x|z) = \frac{p(x; z|x)p(x)}{p(z)}} \tag{2.5}$$

2.4 Estimators

2.4.1 Introduction

As already mentioned, even if we know $p(x|z)$ thanks to equation (2.5), we need more likely a *good* estimation $\hat{x}$ of x. But how can we deduce $\hat{x}$ from $p(x|z)$? As an example we obtained $p(x|z)$ as drawn in figure 2.4, which value should we chose: $\hat{x}_1$, $\hat{x}_2$, other ?

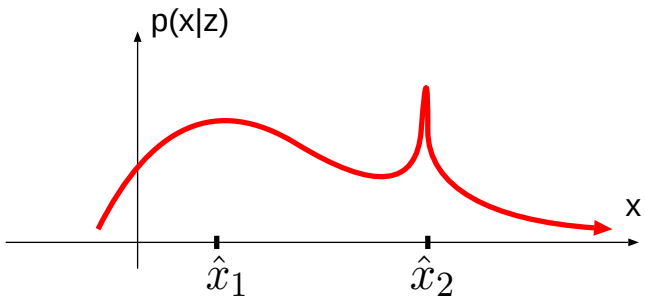


Figure 2.4: Starting from $p(x|z)$ which value $\hat{x}$ should we choose ?

2.4.2 Estimator Bias and Variance

Usually we characterize an estimator $\hat{x}$ of x by:

its bias:

$$B_{\hat{x}} = \mathbf{E}[\hat{x} - x] = \mathbf{E}[\hat{x}] - x$$

- the bias should be null if possible,
- An estimator having a null bias is **unbiased**.

its variance:

$$\mathrm{Var}[\hat{x}] = \mathbf{E}\left[(\hat{x} - x)^2\right] = \mathbf{E}\left[\hat{x}^2\right] - \mathbf{E}[x]^2$$

- The variance should be as small as possible,
- The minimum variance of an estimation problem can theoretically be known: it is the **CRLB** (Cramer-Rao Lower Bound) [22],
- The best estimator for a given estimation problem is *unbiased* and has a variance given by the CRLB.

2.5 Usual estimators

2.5.1 MAP estimator

The **MAP** (Maximum A Posteriori) estimator is rather simple. The best value $\hat{x}_{MAP}$ is the value x such as $p(x|z)$ reaches its maximum (figure 2.5):

$$\boxed{\hat{x}_{MAP} = \arg\max_x p(x|z)} \tag{2.6}$$

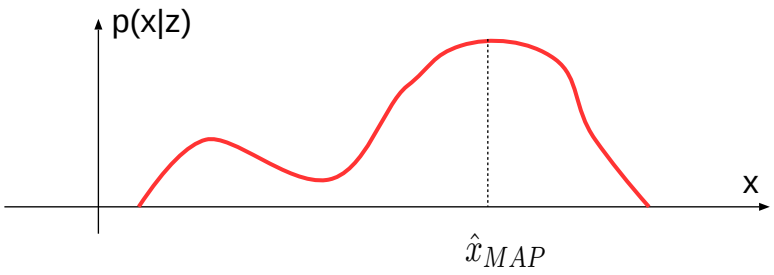


Figure 2.5: MAP estimator

2.5.2 MMSE estimator

The main principle of the **MMSE** (Minimum Mean-Square Error) is to minimize the square errors sum.

$$\hat{x}_{MMSE} = \arg\min_{\hat{x}} \mathbf{E}\left[(\hat{x}-x)^{\top}(\hat{x}-x)|z\right] \tag{2.7}$$

We can demonstrate that :

$$\boxed{\hat{x}_{MMSE} = \mathbf{E}[x|z] = \int x p(x|z) dx}$$

Figure 2.6 illustrates this estimator.

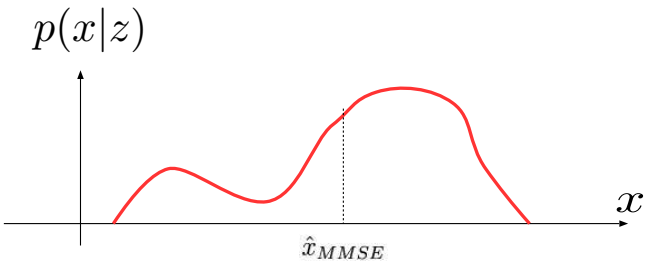


Figure 2.6: MMSE estimator

2.5.3 Bayesian estimators: conclusion

We can meet other estimators in the literature but as usual, none is optimal whatever $p(x|z)$. As a guideline we can notice:

- The behaviour of most of the estimators is approximately the same for symmetrical distributions $p(x|z)$.,
- In the case of multi-modal distributions, the behaviour can lead to strong errors.
- Usually the MAP estimator can be easily numerically computed,
- In the case of the MAP and the MMSE, the denominator $p(z)$ of equation (2.5) does no longer matter since it doesn't depend on x. So for example for the MAP estimator, instead of using equation (2.6) we can use the following:

$$\hat{x}_{MAP} = \arg\max_{x} p(x|z) = \arg\max_{x} \{p(x;z|x).p(x)\} \tag{2.8}$$

2.6 Exercise

We consider a scalar measurement z such as $z = \theta + w$ with: w: noise such as $w \sim \mathcal{N}(0, \sigma_w^2)$ and with the prior knowledge on $\theta \sim \mathcal{N}(\theta_0, \sigma_\theta^2)$. Determine the Bayesian estimator $\hat{\theta}$ of θ.

Solution

We need first to compute $p(z|\theta)p(\theta)$. Actually, we'll need to find its maximum on θ, this means that we are looking for $p(\theta; z|\theta)p(\theta)$.

Since $z = \theta + w$ and $w \sim \mathcal{N}\left(0, \sigma_w^2\right)$, we therefore have $z \sim \mathcal{N}\left(\theta, \sigma_w^2\right)$:

$$p(z|\theta) = \mathcal{N}\left(\theta, \sigma_w^2\right)$$

and so:

$$p(\theta; z|\theta) = \mathcal{N}\left(\theta, \sigma_w^2\right)$$

We also know:

$$p(\theta) = \mathcal{N}(\theta_0, \sigma_\theta^2)$$

The product of two Gaussian functions $\mathcal{N}(\mu_1, \mathbf{C}_1)$ and $\mathcal{N}(\mu_2, \mathbf{C}_2)$ will be a Gaussian function (see § 8.2.6) given by:

$$\mathcal{N}(\mu_1, \mathbf{C}_1) \times \mathcal{N}(\mu_2, \mathbf{C}_2) = k\mathcal{N}(\mu, \mathbf{C})$$

With:

$$\mathbf{C} = \left[\mathbf{C}_1^{-1} + \mathbf{C}_2^{-1}\right]^{-1}$$

$$\bar{} = \left[\mathbf{C}_1^{-1} + \mathbf{C}_2^{-1}\right]^{-1}\left[\mathbf{C}_1^{-1}\mu_1 + \mathbf{C}_2^{-1}\mu_2\right]$$

So $p(\theta; z|\theta)p(\theta)$ will be an un-normalized Gaussian function:

$$p(\theta; z|\theta)p(\theta) \propto \mathcal{N}(\mu, \sigma^2) \propto \mathcal{N}(z, \sigma_w^2) \times \mathcal{N}(\theta_0, \sigma_0^2)$$

With:

$$\sigma^2 = \left[\sigma_w^{-2} + \sigma_\theta^{-2}\right]^{-1} = \frac{1}{\sigma_w^{-2} + \sigma_\theta^{-2}} = \frac{\sigma_w^2.\sigma_\theta^2}{\sigma_w^2 + \sigma_\theta^2}$$

$$\mu = \left[\sigma_w^{-2} + \sigma_\theta^{-2}\right]^{-1}\left[\sigma_w^{-2}z + \sigma_0^{-2}\theta_0\right] = \frac{\sigma_\theta^2 z + \sigma_w^2\theta_0}{\sigma_w^2 + \sigma_\theta^2}$$

Since the result is a Gaussian function, taking the MAP or the MMSE will lead to the same result:

$$\hat{\theta} = \mu = \frac{\sigma_\theta^2 z + \sigma_w^2\theta_0}{\sigma_w^2 + \sigma_\theta^2}$$

Chapter 3

Dynamic Estimation

3.1 Introduction

The goal of the dynamic estimation is to provide an estimation of the **state vector**[1] of a given system. The problem here is much more complicated since, in addition to achieve the estimation, we have to take into account the evolution on this state vector.

We saw the Bayes inference in the previous chapter (see equation (2.4) in §2.3.2). This equation (reminded in eq. (3.1)) uses $p(x)$ which is actually the *prior* knowledge we have on x.

$$p(x|z) = \frac{p(x;z|x)p(x)}{p(z)} \tag{3.1}$$

The dynamic estimation will take advantage of that term $p(x)$ that will stem from the previous estimation. We'll use the following notations:

- x_k: real state vector for time k,
- z_k: measurement achieved for time k
- Z_k: set of measurements achieved until time k. So:

$$Z_k = \{z_0, \cdots, z_k\} \qquad \text{and} \qquad Z_{k-1} = \{z_0, \ldots, z_{k-1}\}$$

Actually we wish to know x_k taking benefit of all measurements z_k we got, so we are looking for $p(x_k|Z_k)$. We can write:

$$p(x_k|Z_k) = \frac{p(x_k;Z_k|x_k) \times p(x_k)}{p(Z_k)} \tag{3.2}$$

We'll see in §3.3 how we can deal with this equation but we first need to talk about *dynamic state formulation*.

3.2 Stochastic state equations

We borrow here eq. (3.3) from the state systems theory which is very well adapted to our problem:

$$\begin{cases} x_k &= f(x_{k-1}, u_k, w_k) \\ z_k &= h(x_k, v_k) \end{cases} \tag{3.3}$$

We have two equations:

[1] We talk about **state** when it describes the main components of a dynamic system

- **Evolution equation**: that defines how the state vector evolves with time and with input u_k. Of course, this prediction is not perfect (otherwise no need to estimate anything !) and a **noise evolution** w_k is added to model that.

  Since x_k is linked to x_{k-1} but not directly to x_{k-2} we use here the **One order Markovian assumption**.

- **Measurement equation**: this equation is actually the one we saw in eq. (2.1) linking z measurement to x_k state. v_k is the *observation noise*.

As we already seen in § 2.3, in order to take benefit of the Bayesian formulation, we prefer to use *pdf* representation of both equations in system (3.3) and we easily derive this system to eq. (3.4):

$$\begin{cases} x_k &= f(x_{k-1}, u_k, w_k) \\ z_k &= h(x_k, v_k) \end{cases} \Rightarrow \begin{cases} p(x_k|x_{k-1}) \\ p(x_k; z_k|x_k) \end{cases} \tag{3.4}$$

Exercise: Consider the linear state model defined by system (3.5):

$$\begin{cases} x_k &= \mathbf{A}x_{k-1} + \mathbf{B}u_k + w_k \\ z_k &= \mathbf{C}x_k + v_k \end{cases} \tag{3.5}$$

$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are constant matrices and v_k, w_k centred Gaussian noises with covariance matrices: $\mathbf{Cov}[v_k] = \mathbf{R}_k$ and $\mathbf{Cov}[w_k] = \mathbf{Q}_k$. Give $p(x_k|x_{k-1})$ and $p(x_k; z_k|x_k)$.

3.3 Equations for Dynamic Bayesian Estimation

We had the following equation:

$$p(x_k|Z_k) = \frac{p(x_k; Z_k|x_k) \times p(x_k)}{p(Z_k)} \tag{3.6}$$

Our goal is now to provide $p(x_k|Z_k)$ using the previous estimation $p(x_{k-1}|Z_{k-1})$ in order to both use all the measurements Z_k but also to avoid to increase the computational cost after each time k.

We need therefore to modify eq. (3.6) in order to take into account not $p(x)$ but this previous estimation $p(x_{k-1}|Z_{k-1})$.

Actually we'll split the problem in two parts: **Prediction** and **Updating**.

3.3.1 Prediction

Starting from $p(x_{k-1}|Z_{k-1})$ we'll try to obtain $p(x_k|Z_{k-1})$: this is the prediction of x_k without the last measurement z_k.

For two random variables x and y we have:

$$p(x) = \int_{-\infty}^{+\infty} p(x, y)dy \qquad \text{and} \qquad p(x, y) = p(x|y)p(y)$$

So:

$$p(x_k|Z_{k-1}) = \int p(x_k, x_{k-1}|Z_{k-1})dx_{k-1}$$

Hence:

$$p(x_k|Z_{k-1}) = \int p(x_k|x_{k-1}, Z_{k-1})p(x_{k-1}|Z_{k-1})dx_{k-1}$$

Since all the past of x_k is stored in x_{k-1} (*Markovian assumption*), Z_{k-1} doesn't bring any news to x_k, so:

$$\boxed{p(x_k|Z_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Z_{k-1})dx_{k-1}} \tag{3.7}$$

This the **Chapman-Kolmogorov** relation [30]. We can notice this equation provides the prediction x_k having all the data until $k-1$ and taking into account:

- the *evolution model* $p(x_k|x_{k-1})$ (see eq. (3.4),
- the **previous estimation** $p(x_{k-1}|Z_{k-1})$ in a **recursive** process as we wanted.

3.3.2 Updating

Having $p(x_k|Z_{k-1})$ we'll try to get $p(x_k|Z_k)$: here we'll take into account the last measurement z_k to provide the wished *pdf* $p(x_k|Z_k)$. Since $Z_k = \{z_0, \cdots, z_k\}$ and $Z_{k-1} = \{z_0, \ldots, z_{k-1}\}$, we'll have:

$$p(x_k|Z_k) = p(x_k|z_0, \ldots, z_k) = \frac{p(z_k|x_k, z_0, \ldots, z_{k-1})p(x_k|z_0, \ldots, z_{k-1})}{p(z_k|z_0, \ldots, z_{k-1})}$$

We need to make here a strong assumption: **all the measurements z_k are statistically independent**. Then:

$$p(z_k|x_k, z_0, \ldots, z_{k-1}) = p(z_k|x_k) \qquad \text{and} \qquad p(z_k|z_0, \ldots, z_{k-1}) = p(z_k)$$

So:

$$p(x_k|z_0, \ldots, z_k) = \frac{p(z_k|x_k)p(x_k|z_0, \ldots, z_{k-1})}{p(z_k)} \propto p(z_k|x_k)p(x_k|z_0, \ldots, z_{k-1})$$

And so:

$$p(x_k|Z_k) \propto p(z_k|x_k)p(x_k|Z_{k-1})$$

Since we'll need to optimize this equation regarding x_k we'll need to use the likelihood function $p(x_k; z|x_k)$ and therefore:

$$\boxed{p(x_k|Z_k) \propto p(x_k; z_k|x_k)p(x_k|Z_{k-1})} \tag{3.8}$$

This equation makes it possible to update the estimation having the last measurement z_k and using the prediction equation (3.7). In a same way eq. (3.8) will be the input pour next time $k+1$ equation (3.7).

Hence, we obtain a **recursive** algorithm that has a constant computational load as depicted in figure 3.1.

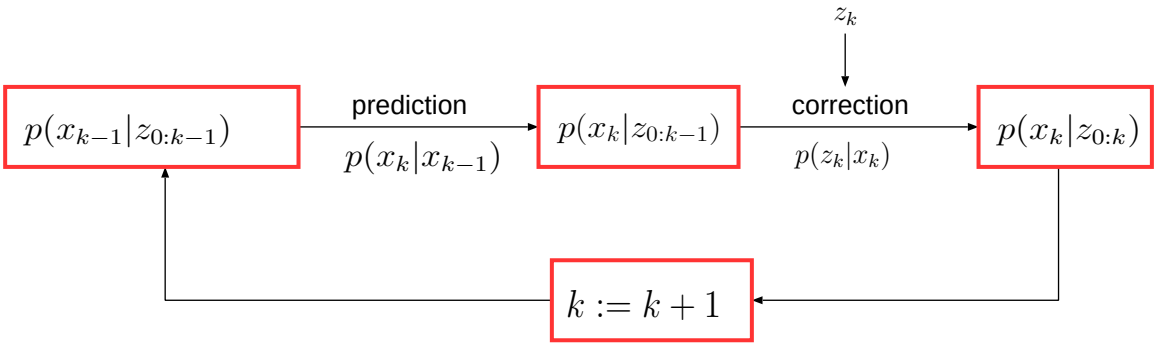


Figure 3.1: Bayesian Dynamic Estimation Algorithm

3.4 Bayesian Estimation: subsidiary remarks

Thanks to eq. (3.7) and eq. (3.8) we have solved the Bayesian Dynamic Estimation. Notice the following points:

- We made two assumptions: 1) the *one order Markov assumption*, that can always be satisfied by choosing a suitable state vector x and 2) the independence of the random noises v_k and w_k (and so the white property of these noises). This last assumption can be an issue however,
- Relationships (3.7) and (3.8) provide *pdf* $p(x_k|Z_{k-1})$ and $p(x_k|Z_k)$. It is mandatory to apply an estimator to provide the estimation $\hat{x}_k$ of x_k (see § 2.4),
- Unfortunately eq. (3.7) and (3.8) are most of the time not tractable (in particular eq.(3.7)),
- Making a linear assumption for f and h functions in eq (3.3) and white, independent and Gaussian assumptions for w_k and v_k noises will provide a tractable solution: this is the **Kalman Filter**.

Chapter 4

Kalman filtering

As seen in §3.4, the main goal of the Kalman filter is to provide a solution to dynamic estimation equations (3.7) and (3.8) assuming several hypothesis.

The Kalman filter has been developed by Kalman [17] for discrete time then by Kalman and Bucy [18] for continuous time. A more detailed presentation can be found in [12].

The Kalman filter follows the same steps than the generic Bayesian Estimation:

- **Prediction** of the future state value,
- **Estimation** of the current state value.

4.1 Linear Stochastic Modelling

The general equations of a stochastic state system is reminded in eq. (4.1).

$$\begin{cases} x_k &= f(x_{k-1}, u_k, w_k) \\ z_k &= h(x_k, v_k) \end{cases} \tag{4.1}$$

Since the main assumption of the Kalman filter relies on the linearity of both functions f and h in (4.1), we'll assume now the following linear state system:

$$\begin{cases} x_{k+1} &= \mathbf{A}x_k + \mathbf{B}u_k + w_k \\ z_k &= \mathbf{C}x_k + v_k \end{cases} \tag{4.2}$$

With:

- x_k, x_{k-1}: State vector for times k and $k-1$,
- u_k: input vectors,
- w_k: additive state noise vector,
- z_k: measurement at time k,
- v_k: noise measurement vector.

v_k and w_k noises are supposed to be **Gaussian**, **white** and **uncorrelated**:

$$p(w_k) = \mathcal{N}(0, \mathbf{Q}_k)$$
$$p(v_k) = \mathcal{N}(0, \mathbf{R}_k)$$

- $\mathbf{Q}_k$ and $\mathbf{R}_k$ are the covariance matrices associated to these noises.
- The **initial state vector** x_0 is assumed to follow also a Gaussian law, (as well as the next steps: Gaussian properties): $\mathcal{N}(\bar{x}_0, \mathbf{P}_0)$ such as:

$$\bar{x_0} = E[x_0] \qquad \text{and} \qquad \mathbf{P}_0 = E[(x_0 - \bar{x_0})(x_0 - \bar{x_0})^\top]$$

4.2 Notations

Consider a random matrix $\mathbf{Z}$ defined for time k as $\mathbf{Z}_k$, for which we want to estimate the value. We name $\mathbf{Z}_{k|l}$ the estimation of $\mathbf{Z}_k$ for time k having measurements until time l. We'll have three possible cases:

- if $k > l$, then $\mathbf{Z}_{k|l}$ will be the *prediction* of $\mathbf{Z}_k$,
- if $k = l$, then $\mathbf{Z}_{k|l}$ will be an *estimation* of $\mathbf{Z}_k$,
- if $l > k$: this is a fitting procedure

In our case we use the following notations:

- x_k: real value of x for time k,
- $x_{k|k}$: estimation of x_k taking into account measurements until k,
- $x_{k|k-1}$: prediction of x_k taking into account measurements until $k-1$,
- $\mathbf{P}_{k|k} = E[(x_{k|k} - x_k)(x_{k|k} - x_k)^\top]$: *a posteriori* covariance matrix on estimation vector $x_{k|k}$,
- $\mathbf{P}_{k|k-1} = E[(x_{k|k-1} - x_k)(x_{k|k-1} - x_k)^\top]$: *a priori* covariance matrix on prediction vector $x_{k|k-1}$.

4.3 Linear Kalman Filter algorithm

The algorithm of the Kalman filter is the following:

1. **Initialization**: we assume to have a Gaussian estimation on the initial state vector value *pdf* : $p(x_0)$
$$p(x_k) = \mathcal{N}(x_{k|k}, \mathbf{P}_{k|k}) \text{ for } k = 0$$
2. $k = k+1$
3. **Prediction**: we predict the Gaussian *pdf* for the next time k:
$$p(x_k|y_0, \ldots, y_{k-1}) = \mathcal{N}(x_{k|k-1}, \mathbf{P}_{k|k-1})$$
4. **Updating**: update the state estimation having the measurement y_k
$$p(x_k|y_0, \ldots, y_k) = \mathcal{N}(x_{k|k}, \mathbf{P}_{k|k})$$
5. goto 2

4.4 Linear Kalman Filter equations

4.4.1 Evolution Equation

We wish to compute $p(x_k|y_0, \cdots, y_{k-1}) = \mathcal{N}\left(x_{k|k-1}, \mathbf{P}_{k|k-1}\right)$, but, since we assume a Gaussian law, we only have to compute $x_{k|k-1}$ and $\mathbf{P}_{k|k-1}$. The solution is (see [17, 18, 3]):

$$x_{k|k-1} = \mathbf{A}x_{k-1|k-1} + \mathbf{B}u_k$$

And:

$$\mathbf{P}_{k|k-1} = \mathbf{A}\mathbf{P}_{k-1|k-1}\mathbf{A}^\top + \mathbf{Q}_{k-1}$$

Notice these relationships are **recursive**: only the $k-1$ estimations matter.

4.4.2 Updating Equation

We wish now to estimate the Gaussian law $p(x_k|y_0,\cdots,y_k)=\mathcal{N}\left(x_{k|k},\mathbf{P}_{k|k}\right)$, so we only need to compute $x_{k|k}$ and its covariance matrix $\mathbf{P}_{k|k}$. The solution is:

- Kalman gain:
$$\mathbf{K}_k=\mathbf{P}_{k|k-1}\mathbf{C}^\top\left[\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^\top+\mathbf{R}_k\right]^{-1}$$
- State vector $x_{k|k}$:
$$x_{k|k}=x_{k|k-1}+\mathbf{K}_k(z_k-\mathbf{C}x_{k|k-1})$$
- Covariance matrix $\mathbf{P}_{k|k}$:
$$\mathbf{P}_{k|k}=(\mathbf{I}-\mathbf{K}_k\mathbf{C})\,\mathbf{P}_{k|k-1}$$

4.4.3 Linear Kalman Filter Equations

The equations are summarized here:

$$\left\{\begin{array}{lcl} x_{k|k-1} &=& \mathbf{A}x_{k-1|k-1}+\mathbf{B}u_k \\ \mathbf{P}_{k|k-1} &=& \mathbf{A}\mathbf{P}_{k-1|k-1}\mathbf{A}^\top+\mathbf{Q}_k \\ x_{k|k} &=& x_{k|k-1}+\mathbf{K}_k(z_k-\mathbf{C}x_{k|k-1}) \\ \mathbf{P}_{k|k} &=& (\mathbf{I}-\mathbf{K}_k\mathbf{C})\mathbf{P}_{k|k-1} \\ \mathbf{K}_k &=& \mathbf{P}_{k|k-1}\mathbf{C}^\top(\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^\top+\mathbf{R}_k)^{-1} \end{array}\right. \tag{4.3}$$

Remark: $(z_k-\mathbf{C}x_{k|k-1})$ is called *Innovation*, and this term is weighted by the *Kalman gain* $\mathbf{K}$,

4.5 Exercise

We wish to estimate the pose and speed of a vehicle running on a road. This vehicle is seen by a camera embedded in a satellite. From the images we can extract the position of the vehicle with a 2 pixels standard deviation Gaussian error. The position (u,v) in the image is linked to the vehicle position (x,y) by:

$$u=e_u\frac{x}{h}\text{ and }v=e_v\frac{y}{h}$$

where h is the satellite altitude and e_u and e_v are projection constants.

We consider a sampling time $T_s=1s$ and the vehicle speed is approximately constant (but we tolerate a 1km/h/s standard deviation Gaussian error). We therefore assume a *constant velocity* model (see Appendix § 8.3.2).

Determine the Kalman filter parameters allowing to solve this problem.

4.5.1 Evolution

We have : $\underline{X}=(x,\dot{x},y,\dot{y})^\top$ and a constant speed model, so :

$$\underline{x}_{k+1}=\begin{pmatrix}1&T_s&0&0\\0&1&0&0\\0&0&1&T_s\\0&0&0&1\end{pmatrix}\begin{pmatrix}x\\\dot{x}\\y\\\dot{y}\end{pmatrix}+\begin{pmatrix}\underline{w}_{kx}\\\underline{w}_{ky}\end{pmatrix}\text{ with }\underline{w}_{kx}=\begin{pmatrix}w_{kx}\\w_{k\dot{x}}\end{pmatrix}\text{ and }\underline{w}_{ky}=\begin{pmatrix}w_{ky}\\w_{k\dot{y}}\end{pmatrix}$$

Noises on x and y are assumed to be uncorrelated, so we have :

$$\mathbf{Q}=\begin{pmatrix}\mathbf{C}_{kx}&\mathbf{0}\\\mathbf{0}&\mathbf{C}_{ky}\end{pmatrix}\qquad\text{with}\quad\mathbf{C}_{kx}=\mathbf{C}_{ky}=\sigma_w^2\begin{pmatrix}\frac{T_s^3}{3}&\frac{T_s^2}{2}\\\frac{T_s^2}{2}&T_s\end{pmatrix}$$

With $\sigma_w=1km/h/s=\frac{1000}{3600}=0.27m/s^2$.

4.5.2 Updating

We know that :

$$\hat{u} = e_u \frac{x}{h} + \epsilon_u \qquad \text{and} \qquad \hat{v} = e_v \frac{y}{h} + \epsilon_v$$

We will have :

$$z = \begin{pmatrix} \hat{u} \\ \hat{v} \end{pmatrix} = \mathbf{C}\underline{X} + \underline{v}_k \qquad \text{with} \qquad \underline{v}_k = \begin{pmatrix} \epsilon_u \\ \epsilon_v \end{pmatrix}$$

And :

$$\mathbf{C} = \begin{pmatrix} \frac{e_u}{h} & 0 & 0 & 0 \\ 0 & 0 & \frac{e_v}{h} & 0 \end{pmatrix}$$

Since $\hat{u}$ and $\hat{v}$ are assumed to be uncorrelated and having a 2 pixels noise, we can write :

$$\mathbf{R} = \begin{pmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{pmatrix} = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}$$

4.6 Linear Kalman filter: remarks

Regarding the Linear Kalman Filter, we can notice the following points:

- In the case where the Gaussian hypothesis is not verified, the filter provides a sub-optimal estimation,
- The Kalman Filter can cope with non-stationnary noises, indeed, in these cases, $\mathbf{Q}_k$ and $\mathbf{R}_k$ matrices changes over time but the equations uses the current value. No restriction is so required here,
- However the **white noise constraint** (on both v_k and w_k) can be a problem prone to lead to over-convergences. Several solutions exist, such as the *covariance intersection* method [13] for example (see also § 6.3.3),
- The computational cost of the filter is due mainly to the matrix inversion (size $N \times N$, N is z size),
- It is possible to reduce the complexity by splitting this vector in several sets of uncorrelated components (see § 6.2.2),
- The linearity constraint is probably the main issue of the Linear Kalman filter,
- Several approaches exist:
  - Extented Kalman filter (EKF),
  - Uncented Kalman Filter (UKF),
  - Etc...

The next chapter is dedicated to Extended Kalman filter (EKF).

Chapter 5

Extented Kalman Filter

5.1 Introduction

In order to solve the estimation problem even in non-linear cases we encounter most of the time, the classical solution is to *linearize* the state equations. Let's turn back to the generic stochastic state system:

$$\begin{cases} x_k &= f(x_{k-1}, u_k, w_k) \\ z_k &= h(x_k, v_k) \end{cases} \tag{5.1}$$

Since h and f are nonlinear functions, matrices **A**, **B** and **C** in eq. (3.5) disappear. Our problem is now to deal with this new issue.

5.2 Non linear functions mean and variance

Consider a random value x with expected value $\mathbf{E}[x]$ and with variance $\mathbf{Var}[x] = \sigma_x^2$ and the following scalar equation[1]

$$y = f(x)$$

What are the expected value and the variance of y ?

Expected value of y: We can write $x = \mu_x + \epsilon_x$ with $\mu_x = \mathbf{E}[x]$ and ϵ_x is stochastic part of x such as $\mathbf{E}[\epsilon_x] = 0$ and $\mathbf{Var}[\epsilon_x] = \sigma_x^2$. We have (see appendix § 8.1.2):

$$y = f(\mu_x + \epsilon_x) \approx f(\mu_x) + \epsilon_x \frac{\partial f}{\partial x}\Big|_{x=\mu_x}$$

So:

$$\mathbf{E}[y] \approx \mathbf{E}[f(\mu_x)] + \mathbf{E}\left[\epsilon_x \frac{\partial f}{\partial x}\right] = \mathbf{E}[f(\mu_x)] + \mathbf{E}[\epsilon_x]\frac{\partial f}{\partial x}$$

$$\mathbf{E}[y] \approx f(\mu_x)$$

Variance: The variance of $y = f(x)$ will be calculated by:

$$\mathbf{Var}[y] = \mathbf{Var}[f(x)] \approx \mathbf{Var}\left[f(\mu_x) + \epsilon_x \frac{\partial f}{\partial x}\Big|_{x=\mu_x}\right] = \mathbf{Var}\left[\epsilon_x \frac{\partial f}{\partial x}\right]$$

$$\mathbf{Var}[y] \approx \mathbf{Var}[\epsilon_x]\left(\frac{\partial f}{\partial x}\right)^2 = \sigma_x^2\left(\frac{\partial f}{\partial x}\right)^2$$

[1] For a recall on random values and their properties, the reader should read § 8.1.2 and § 8.1.3.

5.2.1 Non-linear functions: vectorial case

Consider a random vectorial value $x = (x_1, \cdots, x_N)^\top$ with expected value $\mu_x = \mathbf{E}[x]$ and with covariance $\mathbf{Cov}[x] = \mathbf{C}_x$, and the following vectorial equation:

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_M \end{pmatrix} = f(x) = \begin{pmatrix} f_1(x_1, \cdots, x_N) \\ \vdots \\ f_M(x_1, \cdots, x_N) \end{pmatrix}$$

What are the expected value and the covariance of y ?

Expected value of y: We can write $x = \mu_x + \epsilon_x$.

Here, ϵ_x is stochastic part of x such as $\mathbf{E}[\epsilon_x] = 0$ and $\mathbf{Cov}[\epsilon_x] = \mathbf{C}_x$. We have

$$y = f(\mu_x + \epsilon_x) \approx f(\mu_x) + \frac{\partial f}{\partial x}\Big|_{x=\mu_x} \epsilon_x$$

With the **Jacobian Matrix** $\mathbf{J}_{fx}$:

$$\mathbf{J}_{fx} = \frac{\partial f}{\partial x}\Big|_{x=\mu_x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{pmatrix}$$

So we have:

$$y \approx f(\mu_x) + \mathbf{J}_{fx}\epsilon_x$$

So:

$$\mathbf{E}[y] \approx \mathbf{E}[f(\mu_x)] + \mathbf{J}_{fx}\mathbf{E}[\epsilon_x] = f(\mu_x)$$

The Covariance $\mathbf{C}_y$ of y will be (since $f(\mu_x)$ is a constant):

$$\mathbf{Cov}[y] \approx \mathbf{Cov}[\mathbf{J}_{fx}\epsilon_x] = \mathbf{J}_{fx}\mathbf{Cov}[\epsilon_x]\mathbf{J}_{fx}^\top$$

So finally:

$$\mathbf{Cov}[y] = \mathbf{C}_y \approx \mathbf{J}_{fx}\mathbf{C}_x\mathbf{J}_{fx}^\top$$

If $z = f(x, y)$ with x and y are **independent** random values, then:

$$\mathbf{E}[z] \approx \mathbf{f}(\mu_x, \mu_y) \qquad \text{and} \qquad \mathbf{Cov}[z] \approx \mathbf{J}_{fx}\mathbf{C}_x\mathbf{J}_{fx}^\top + \mathbf{J}_{fy}\mathbf{C}_y\mathbf{J}_{fy}^\top$$

5.3 Linearized Kalman Filter Equations

Let's use the previous and use that to update the Linear Kalman equations given in eq (4.3). Remember the prediction equation is given by: $x_k = f(x_{k-1}, u_k, w_k)$. We have two cases:

- If input vector u_k is perfectly known,
  - the best prediction of x_k will be:

$$x_{k|k-1} = f(x_{k-1|k-1}, u_k, 0)$$

  - The covariance matrix $\mathbf{P}_{k|k-1}$ of $x_{k|k-1}$ will be:

$$\begin{aligned} \mathbf{P}_{k|k-1} &= \mathbf{J}_{fX}Cov[x_{k-1|k-1}]\mathbf{J}_{fX}^\top + \mathbf{J}_{fw}Cov[w_k]\mathbf{J}_{fw}^\top \\ &= \mathbf{J}_{fX}\mathbf{P}_{k-1|k-1}\mathbf{J}_{fX}^\top + \mathbf{J}_{fw}\mathbf{Q}_k\mathbf{J}_{fw}^\top \end{aligned}$$

- If input vector u_k is only given by a measuremement $\hat{u}_k$ with centered noise with covariance $\mathbf{C}_{uk}$ then $u_k \sim \mathcal{N}(\hat{u}_k, \mathbf{C}_{uk})$ and:
  - the best prediction of x_k will be:

$$x_{k|k-1} = f(x_{k-1|k-1}, \hat{u}_k, 0)$$

  - The covariance matrix $\mathbf{P}_{k|k-1}$ of $x_{k|k-1}$ will be:

$$\begin{aligned} \mathbf{P}_{k|k-1} &= \mathbf{J}_{fX} Cov[x_{k-1|k-1}] \mathbf{J}_{fX}^\top + \mathbf{J}_{fu} \mathbf{C}_{uk} \mathbf{J}_{fu}^\top + \mathbf{J}_{fw} Cov[w_k] \mathbf{J}_{fw}^\top \\ &= \mathbf{J}_{fX} \mathbf{P}_{k-1|k-1} \mathbf{J}_{fX}^\top + \mathbf{J}_{fu} \mathbf{C}_{uk} \mathbf{J}_{fu}^\top + \mathbf{J}_{fw} \mathbf{Q}_k \mathbf{J}_{fw}^\top \end{aligned}$$

5.3.1 Updating equations

We know z_k is given by $z_k = h(x_k, v_k)$, the estimated value $z_{k|k-1}$ will be approximately given by:

$$z_{k|k-1} = h(x_{k|k-1}, 0)$$

$x_{k|k}$ will be:

$$x_{k|k} = x_{k|k-1} + \mathbf{K}_k \left(z_k - h(x_{k|k-1}, 0) \right)$$

In a similar way, we can linearize observation equation around x_0 and 0 for the centered noise v_k:

$$z_k \approx h(x_0, 0) + \mathbf{J}_{hX}(x_{k-1|k-1} - x_0) + \mathbf{J}_{hv} v_k$$

$\mathbf{J}_{hX}$ and $\mathbf{J}_{hv}$ are the *Jacobian* matrices of fonction h:

$$\mathbf{J}_{hX} = \left.\frac{\partial h}{\partial x}\right|_{x=x_0} \text{ and } \mathbf{J}_{hv} = \left.\frac{\partial h}{\partial v}\right|_{v=0}$$

By analogy with linear systems, we have:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{J}_{hX}) \mathbf{P}_{k|k-1}$$

and :

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{J}_{hX}^\top (\mathbf{J}_{hX} \mathbf{P}_{k|k-1} \mathbf{J}_{hX}^\top + \mathbf{J}_{hv} \mathbf{R}_k \mathbf{J}_{hv}^\top)^{-1}$$

5.3.2 Linearized Kalman filter : Equations

We consider the following state system: $\left\{ \begin{array}{lcl} x_k &=& f(x_{k-1}, w_k, u_k) \\ z_k &=& h(x_k, v_k) \end{array} \right.$. The solution of the Linearized Kalman filter will be:

$$\left\{ \begin{array}{lcl} x_{k|k-1} &=& f\left(x_{k-1|k-1}, \hat{u}_k, 0\right) \\ \mathbf{P}_{k|k-1} &=& \mathbf{J}_{fX} \mathbf{P}_{k-1|k-1} \mathbf{J}_{fX}^\top + \mathbf{J}_{fw} \mathbf{Q}_k \mathbf{J}_{fw}^\top + \mathbf{J}_{fu} \mathbf{C}_{u_k} \mathbf{J}_{fu}^\top \\ x_{k|k} &=& x_{k|k-1} + \mathbf{K}_k \left(z_k - h(x_{k|k-1}, 0) \right) \\ \mathbf{P}_{k|k} &=& (\mathbf{I} - \mathbf{K}_k \mathbf{J}_{hX}) \mathbf{P}_{k|k-1} \\ \mathbf{K}_k &=& \mathbf{P}_{k|k-1} \mathbf{J}_{hX}^\top (\mathbf{J}_{hX} \mathbf{P}_{k|k-1} \mathbf{J}_{hX}^\top + \mathbf{J}_{hv} \mathbf{R}_k \mathbf{J}_{hv}^\top)^{-1} \end{array} \right.$$

With:

$$\begin{aligned} \mathbf{J}_{fX} &= \left.\frac{\partial f}{\partial x}\right|_{x=x_0} \quad , \quad \mathbf{J}_{fw} = \left.\frac{\partial f}{\partial w}\right|_{w=0}, \mathbf{J}_{fu} = \left.\frac{\partial f}{\partial u}\right|_{u=\hat{u}} \\ \mathbf{J}_{hX} &= \left.\frac{\partial h}{\partial x}\right|_{x=x_0} \quad , \quad \mathbf{J}_{hv} = \left.\frac{\partial h}{\partial v}\right|_{v=0} \end{aligned}$$

5.4 Extended Kalman Filter

The main problem of the Linearized Kalman Filter is that the linearization is achieved around a **fixed value** x_0. If this value is far from the real one, the linearization leads to errors and even to instabilities of the filter. The Extended Kalman Filter principle is to linearize the state equation (see [26]):

- around the estimated value $x_{k-1|k-1}$ for prediction step,
- around the prior estimated value $x_{k|k-1}$ for updating step,
- around the current measurement $\underline{\hat{u}}_k$

This involves *the Jacobians computation at each time* k.

5.4.1 EKF Equations

Equations of the Extended Kalman Filter are grouped in (5.2).

$$\left\{\begin{array}{lcl} x_{k|k-1} & = & f\left(x_{k-1|k-1}, \underline{\hat{u}}_k, 0\right) \\ \mathbf{P}_{k|k-1} & = & \mathbf{J}_{fX}\mathbf{P}_{k-1|k-1}\mathbf{J}_{fX}^\top + \mathbf{J}_{fu}\mathbf{C}_{uk}\mathbf{J}_{fu}^\top + \mathbf{J}_{fw}\mathbf{Q}_{wk}\mathbf{J}_{fw}^\top \\ x_{k|k} & = & x_{k|k-1} + \mathbf{K}_k\left(z_k - h(x_{k|k-1}, 0)\right) \\ \mathbf{P}_{k|k} & = & (\mathbf{I} - \mathbf{K}_k\mathbf{J}_{hX})\mathbf{P}_{k|k-1} \\ \mathbf{K}_k & = & \mathbf{P}_{k|k-1}\mathbf{J}_{hX}^\top(\mathbf{J}_{hX}\mathbf{P}_{k|k-1}\mathbf{J}_{hX}^\top + \mathbf{J}_{hv}\mathbf{R}_k\mathbf{J}_{hv}^\top)^{-1} \end{array}\right. \tag{5.2}$$

with:

$$\mathbf{J}_{fX} = \left.\frac{\partial f}{\partial x}\right|_{x=x_{k-1|k-1}} \quad , \quad \mathbf{J}_{fu} = \left.\frac{\partial f}{\partial u}\right|_{u=\underline{\hat{u}}_k} \quad , \quad \mathbf{J}_{fw} = \left.\frac{\partial f}{\partial w}\right|_{w=0}$$

$$\mathbf{J}_{hX} = \left.\frac{\partial h}{\partial x}\right|_{x=x_{k|k-1}} \quad , \quad \mathbf{J}_{hv} = \left.\frac{\partial h}{\partial v}\right|_{v=0}$$

5.4.2 Exercice

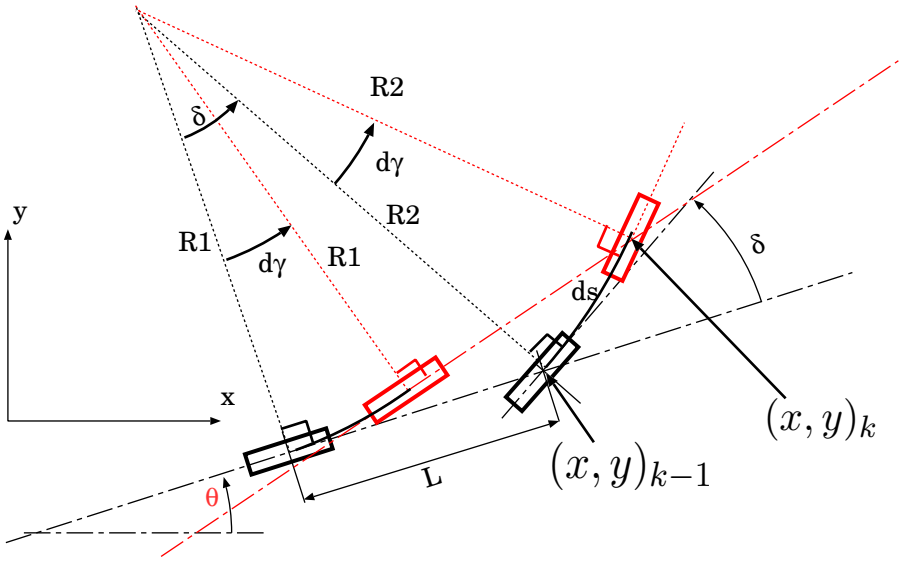
We want to estimate accurately a vehicle position $X = (x, y, \theta)^\top$ by using a GPS receiver, an on-board odometer and a wheel angle sensor.

- GPS receiver provides the vehicle position estimation $(\hat{X}, \hat{y})$ with an error assumed to be stationary, white, Gaussian and having a σ^2 variance.
- The odometer is assumed to provide an estimation $\hat{ds}$ of the vehicle displacement between k and $k+1$ with a white gaussian error with variance σ_{ds}^2.
- The wheel angle δ is given by a sensor that gives an estimation $\hat{\delta}$ of δ with a white gaussian error with variance σ_δ^2.
- We assume the following Ackermann model:

$$\left\{\begin{array}{lcl} x_k & = & x_{k-1} + ds\cos(\theta_{k-1} + d\theta/2) \\ y_k & = & y_{k-1} + ds\sin(\theta_{k-1} + d\theta/2) \\ \theta_k & = & \theta_{k-1} + \frac{ds}{L}\sin(\delta) \end{array}\right. \qquad \text{with} \quad d\theta = \frac{ds}{L}\sin\theta$$

  L is the distance between vehicle axles.

- Determine the EKF parameters to solve the problem.



Solution

Observation equation

- We have

$$x_{gps} = x + v_{xgps} \qquad \text{and} \qquad y_{gps} = y + v_{ygps}$$

- this is a **linear** equation so:

$$z_k = \begin{pmatrix} x_{gps} \\ y_{gps} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}}_{\underline{x_k}} + \underbrace{\begin{pmatrix} v_{xgps} \\ v_{ygps} \end{pmatrix}}_{v_k}$$

- We have also:

$$\mathbf{R} = Cov[v_k] = \begin{pmatrix} \sigma_{gps}^2 & 0 \\ 0 & \sigma_{gps}^2 \end{pmatrix} = \sigma_{gps}^2 \mathbf{I}_{2\times 2}$$

Updating equation

- We look for covariance of $x_{k+1|k}$. We'll have:

$$\mathbf{P}_{k|k-1} = Cov[x_{k|k-1}] = Cov[\mathbf{f}(x_{k-1|k-1}, \hat{ds}, \hat{\delta})] = \mathbf{J}_{fx}\mathbf{P}_{k-1|k-1}\mathbf{J}_{fx}^\top + \mathbf{J}_{ds}\sigma_{ds}^2\mathbf{J}_{ds}^\top + \mathbf{J}_\delta\sigma_\delta^2\mathbf{J}_\delta^\top$$

- $\mathbf{J}_\delta$ and $\mathbf{J}_{ds}$ are the Jacobian matrices of fonction $\mathbf{f}$ with respect to δ and ds.
- They are:

$$\mathbf{J}_{ds} = \begin{pmatrix} \frac{\partial f_x}{\partial ds} \\ \frac{\partial f_y}{\partial ds} \\ \frac{\partial f_\theta}{\partial ds} \end{pmatrix} = \begin{pmatrix} \cos(\theta + \frac{d\theta}{2}) \\ \sin(\theta + \frac{d\theta}{2}) \\ \frac{\sin\delta}{L} \end{pmatrix}$$

$$\mathbf{J}_\delta = \begin{pmatrix} \frac{\partial f_x}{\partial \delta} \\ \frac{\partial f_y}{\partial \delta} \\ \frac{\partial f_\theta}{\partial \delta} \end{pmatrix} = \begin{pmatrix} -\frac{ds^2}{2L}\cos\delta\sin\left(\theta + \frac{d\theta}{2}\right) \\ \frac{ds^2}{2L}\cos\delta\cos\left(\theta + \frac{d\theta}{2}\right) \\ \frac{ds}{L}\cos(\delta) \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \\ \frac{ds}{L}\cos(\delta) \end{pmatrix}$$

Jacobian matrice $\mathbf{J}_{fx}$ is given by:

$$\mathbf{J}_{fx} = \begin{pmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} & \frac{\partial f_x}{\partial \theta} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} & \frac{\partial f_y}{\partial \theta} \\ \frac{\partial f_\theta}{\partial x} & \frac{\partial f_\theta}{\partial y} & \frac{\partial f_\theta}{\partial \theta} \end{pmatrix} = \begin{pmatrix} 1 & 0 & -ds\sin\left(\theta + \frac{d\theta}{2}\right) \\ 0 & 1 & ds\cos\left(\theta + \frac{d\theta}{2}\right) \\ 0 & 0 & 1 \end{pmatrix}$$

EKF equations

$$\begin{cases} x_{k|k-1} &= f\left(x_{k-1|k-1}, \hat{ds}, \hat{\delta}\right) \\ \mathbf{P}_{k|k-1} &= \mathbf{J}_{fx}\mathbf{P}_{k|k}\mathbf{J}_{fx}^\top + \mathbf{J}_{ds}\sigma_{ds}^2\mathbf{J}_{ds}^\top + \mathbf{J}_\delta\sigma_\delta^2\mathbf{J}_\delta^\top \\ x_{k|k} &= x_{k|k-1} + \mathbf{K}_k\left(z_k - \mathbf{C}x_{k|k-1}\right) \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\mathbf{C})\mathbf{P}_{k|k-1} \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{C}^\top(\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^\top + \mathbf{R}_k)^{-1} \end{cases}$$

With:

$$\mathrm{J}_{ds} = \begin{pmatrix} \cos(\theta + \frac{d\theta}{2}) \\ \sin(\theta + \frac{d\theta}{2}) \\ \frac{\sin\delta}{L} \end{pmatrix}, \mathrm{J}_\delta = \begin{pmatrix} -\frac{ds^2}{2L}\cos\delta\sin\left(\theta + \frac{d\theta}{2}\right) \\ \frac{ds^2}{2L}\cos\delta\cos\left(\theta + \frac{d\theta}{2}\right) \\ \frac{ds}{L}\cos(\delta) \end{pmatrix}, \mathrm{C} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \text{ and } d\theta = \frac{ds}{L}\sin\delta$$

5.5 Kalman filter conclusions

The EKF is a very powerful tool, nevertheless:

- If the evolution stage leads to a far state estimation, the filter can diverge, the **Uncented Kalman Filter** (UKF, see [16]) has been developed to reduce this problem by avoiding high jumps of the linearization point.
- The Gaussian assumption is no longer guarantied, indeed, a Gaussian noise remains Gaussian by linear transformation only.
- The multi-modalities are not taken into account at all in the EKF. Several approaches cope with this problem, mainly: Gaussian Mixtures [33, 2] or Particles filters [4, 10].

Chapter 6

Data fusion

6.1 Introduction

The main goal of data fusion is to combine together several measurements (usually coming from several sensors) in order to provide a better result than if we had to do the same with only one sensor [6].

Data fusion is a very large topic. Actually data fusion aims at improving both estimation and decision [15, 34, 1]. Regarding the **estimation**, we would like, for example to get a better localization of a robot using a GNSS receiver (GPS, Glonass, Galileo), detected features in the world (a church, a bridge) that we already have stored in a map, gyro-sensors embedded in the robot, etc.

The **decision** needs to be considered for instance if we wish to give an appropriate choice between destroying or not a rocket after having a warning light on while the other indicators seem indicate all is fine [6].

Another classical issue we encounter in data fusion is the **tracking** problem. In this case we need to combine both precision and decision having multiple incoming data and multiple state to estimate [6], [24]. For example suppose we have manage an autonomous car on an highway. It is worth to identify and track ahead vehicles in order to anticipate their behaviour. To do so, we need to initiate **tracks** for each of them and feed these tracks with new detections. But we can have false associations, managing new tracks, closing dead tracks, etc.

In this section we only concentrate on how we can optimize a state estimation with data coming from several possibly un-synchronized sensors. Decision will not be addressed here (though a short example will be given is §7.1).

According the application, we can meet several architectures. Mainly we can distinguish the centralized an non-centralized fusion architectures.

6.2 Centralized Fusion Architecture

6.2.1 Introduction

The centralized fusion aims at combining in the same process (most of the time on the same processor) the incoming data (see Fig.6.1). Sometimes Estimation and Fusion are grouped in the same block. Usually in the situation presented in the figure, some data can be cancelled (for instance, we can imagine data coming from GPS that are not reliable and a RAIM[1].

We only suppose in the rest of this chapter that the measurements are reliable and need to be fused in the estimation process. For a more detailed review the reader should refer for example to [7].

[1] Receiver Autonomous Integrity Monitoring (RAIM) has cancelled it

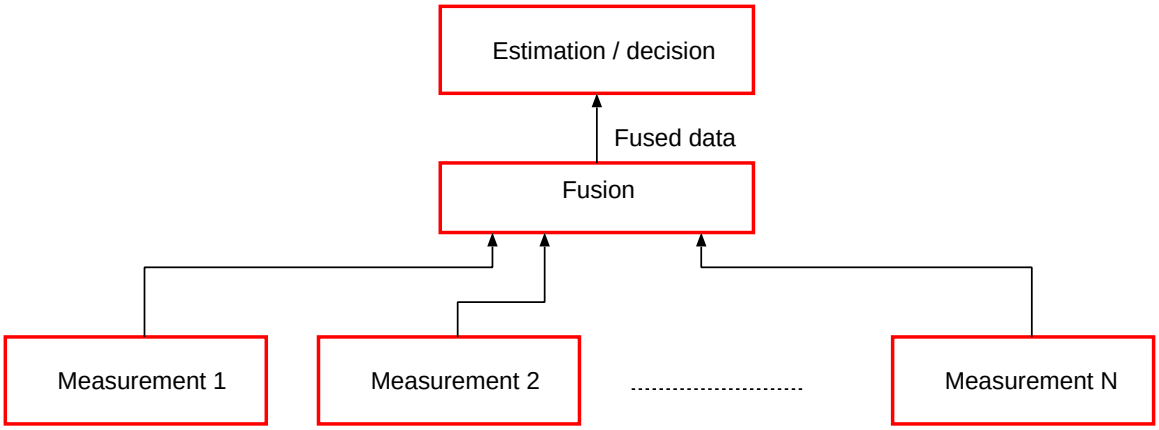


Figure 6.1: Centralized Fusion: Usually Estimation / Fusion are achieved in the same block

6.2.2 Synchronized data

We suppose here to have N incoming measurements z_{ik} with $(i \in [1, N])$ **at the same time** k. Each one of these data z_{ik} is linked to the state vector x_k by the following equation:

$$z_{ik} = h_i(x_k, v_{ik})$$

Here, v_{ik} is the noise peculiar to measurements z_{ik}. Usually the fusion is done with a global Kalman filter. Two approaches can be found.

The **first one** is the simplest: it is to group all the z_{ik} measurements in a same vector $z_k = (z_{1k}, z_{2k}, \cdots, z_{Nk})^\top$.

It will be necessary to define the global covariance matrix $\mathbf{R} = \mathbf{Cov}[z_k]$ (see § 4.4.2):

$$\mathbf{R}_k = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \dots & \mathbf{R}_{1N} \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \dots & \mathbf{R}_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{R}_{N1} & \mathbf{R}_{N2} & \dots & \mathbf{R}_{NN} \end{pmatrix}_k \tag{6.1}$$

The problem becomes now a classical estimation problem and the solution will be given as shown for instance in § 4.4.3 or § 5.4.

In eq. (6.1), $\mathbf{R}_{ij}$ are the covariance links between the data. Usually the data can be assumed as un-correlated, in this case we'll have:

$$\mathbf{R}_k = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{22} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{R}_{NN} \end{pmatrix}_k$$

Figure 6.2 depicts the used algorithm in the Dynamic Bayesian framework. The main advantage of this solution is that we can take into account all the relationships (correlation) between the data, but (as we can see in the Kalman equations, refer to § 5.4.1) the computational load is due to the matrix inversion in the Kalman gain (see eq (5.2)). And, extend vector z_k reduces in $O(N^3)$ the speed-up.

The **second** approach takes advantage of the fact that usually z_{ik} data are uncorrelated. In this case we use the process presented in figure 6.3. The algorithm is the following:

1. State x_k initialization for time k, (i.e $x_{k|k}$ and $\mathbf{P}_{k|k}$)
2. $k \longleftarrow k+1$

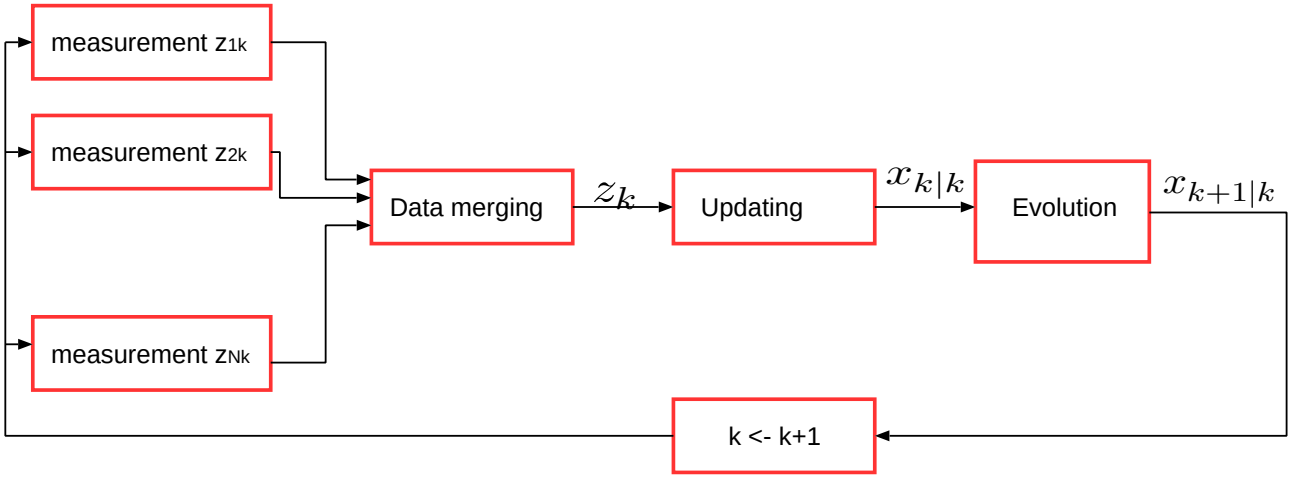


Figure 6.2: Synchronized data fusion

3. Prediction until k (i.e. $x_{k|k-1}$ and $\mathbf{P}_{k|k-1}$ evaluation)
4. For each measurement z_{ik} update state :
   - $x_{k|k_1}$ and $\mathbf{P}_{k|k_1}$ updating with $x_{k-1|k-1}$, $\mathbf{P}_{k-1|k-1}$ and measurement z_{1k},
   - $x_{k|k_2}$ and $\mathbf{P}_{k|k_2}$ updating with $x_{k|k_1}$, $\mathbf{P}_{k|k-1_1}$ and measurement z_{2k},
   - ..
   - $x_{k|k} = x_{k|k_N}$ and $\mathbf{P}_{k|k_N}$ updating with $x_{k|k-1_{N-1}}$, $\mathbf{P}_{k|k-1_{N-1}}$ and measurement z_{Nk},
5. Goto 2

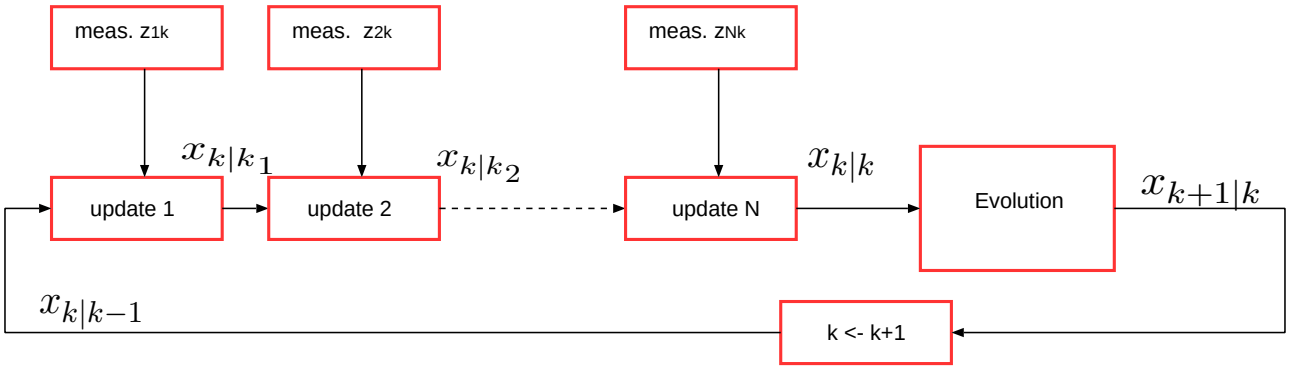


Figure 6.3: Iterated fusion

Remarks: this approach is interesting but it is worth notice the following points:

- This fusion is **optimal** in the first case, otherwise, the covariances between z_{ik} measurements are not taken into account: this can lead to sub-optimal estimation or even **over-convergences** is the related noises are correlated.
- In the first case, spurious measurements and *non-linearities* are smoothed between the whole data set.
- The second case is usually easier to implement, and the computational costs are lower. However no smoothing effect can be done here.
- The main issue to these approaches is that the measurements are required to be **synchronized**. This can be obtained with a prior synchronization step.

6.2.3 Non-synchronized data fusion

Most of the time, z_{ik} measurements come from different sensors are no synchronization can be expected here to use the previous approach.

Moreover, it is more or less assumed that the time between $k-1$ and k is constant (an likely given by a common clock). Actually if a control of the robot is required, the estimation have indeed to be done at a **constant period** T_s that has no link with the sensors clock.

Several solutions exist to solve this issue: data synchronization and delayed processing [34].

6.2.4 Data synchronization

In the first case, all the data are interpolated or extrapolated in order to fit the required sampling time.

This approach is however sup-optimal most of the case because of the extra/interpolation errors and the related noise estimation. Moreover the latency of the data cannot easily be taken into account.

6.2.5 Data fusion "on the fly"

This approach aims at solving both the synchronization and the latency problems [34]. It is worth to distinguish what is *data* and what is *measurement*.

A data d_i is the raw information (an image for instance) taken by the sensor at date t_{di}.

A measurement z_i is the output of a given processing g_i having data d_i as input. z_i goes in the fusion system at date $t_{zi} = t_{di} + \Delta_{ti}$.

We therefore have:

$$z_i = g_i(d_i, \Delta_{ti})$$

Actually, Δ_{ti} represents the latency of measurement z_i, it includes the processing time and the routing time.

Usually the measurement time d_{zi} can be easily known since it is the date the measurement comes in the fusion system. Sometimes t_{di} can be sent by the sensor itself (and Δ_{ti} can be deduced from d_{zi}). But most of the times, we need to make prior times analysis to estimate Δ_{ti} for a given sensor.

Since sensors have different behaviour we'll have to face different latencies and even sometimes difficult situations shown in figure 6.4. For instance, measurement z_5 incomes after z_4 but the corresponding data d_5 has been acquired **before** d_4.

The algorithm presented in [34] is based on an *observation list*. Suppose we want to estimate periodically the global state at each T_i period (for instance to achieve a control correction of the robot trajectory).

For $t = T_0$ (see fig.6.4) we'll have to achieve an evolution step from the last estimation until time T_0 in order to get $\hat{x}_{T0}$.

At time t_{z1} measurement z_1 gets in the fusion system. Its corresponding data d_1 arrived even before (at time t_{d1}), so z_1 measurement is stored in the observation list with t_{d1} time stamp.

At $t = T_1$, we need to achieve an evolution step on the estimated state $\hat{x}_{T0}$ until t_{d1}, update this state with measurement z_1, and achieve a new evolution step until T_1 in order to provide $\hat{x}_{T1}$.

The process is iterated an we always update the estimated state starting from **the last state estimation done before the last not yet processed data**. This allow to solve difficult cases like z_5. Indeed z_5 comes after T_4 for which we already have an estimation $\hat{x}_{T4}$ but the corresponding data d_5 has a related date t_{d5} such as $t_{d5} < t_{t4}$. Here we start from T_3 and the estimation $\hat{x}_{T3}$, we achieve several evolution / updating steps to take into account successively z_5 and z_4 and provide a new estimation $\hat{x}_{T4}$.

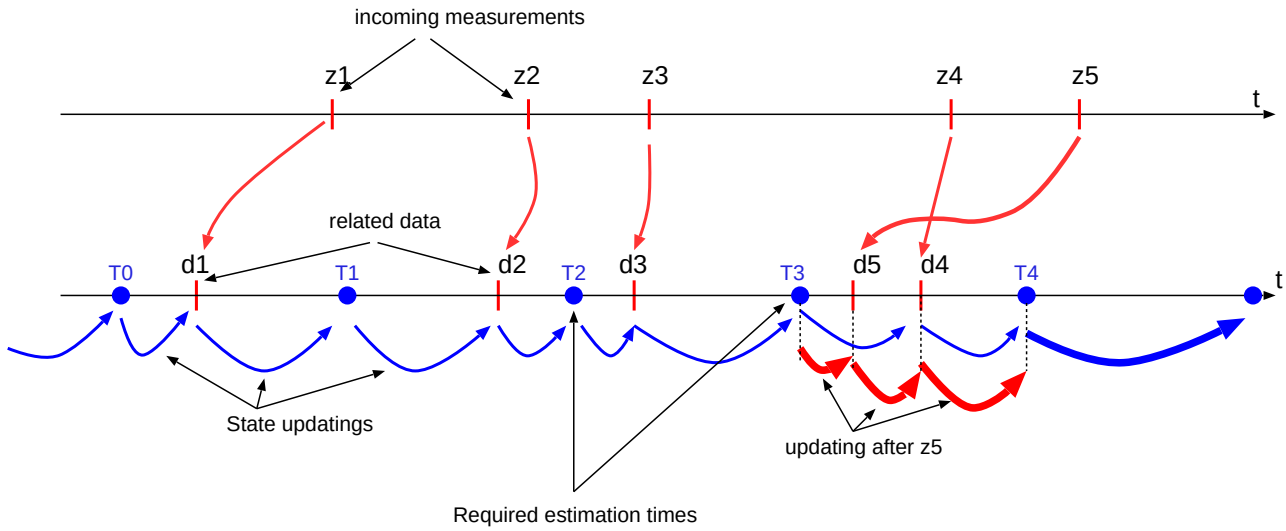


Figure 6.4: Non-synchronized data fusion

6.3 Non-centralized Fusion Architecture

6.3.1 Introduction

It is not always possible to get a centralized fusion. Sometimes it is even better to have un-centralized one. It is actually the case fore a fleet of vehicles were each one needs to take benefit of the measurements done by the others. Even if it is always possible to build a centralized data fusion system (with a global system that communicates with all the vehicles), it is usually better to avoid this global system in order to optimize the safety and the issues coming from the communication losses with this global system [15, 29, 5, 20].

Moreover the scalability of a non-centralized fusion system is better: adding new node becomes simpler (fig 6.5).

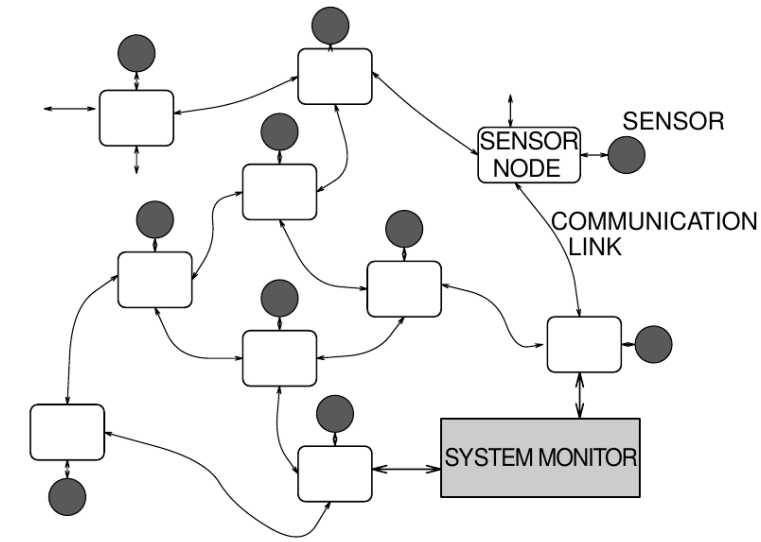


Figure 6.5: Non-centralized fusion with multiple nodes ([15])

A well-known issue is however the **rumour** effect [21, 15]. This problem is depicted in fig 6.6 (from [19]).The first robot R_1 localizes itself and sends its estimated pose to robot R_2. This one sends in turn the received poses of all the robots (only R_1 here) to all the robots in the neighborhood

(only R_1 here). So R_1 can combine this information with its own pose. However since it is the same information it sent, an over-convergence problem appears [19] that usually leads to integrity losses (see §4.6). Several solutions have been proposed to solve this important issue.

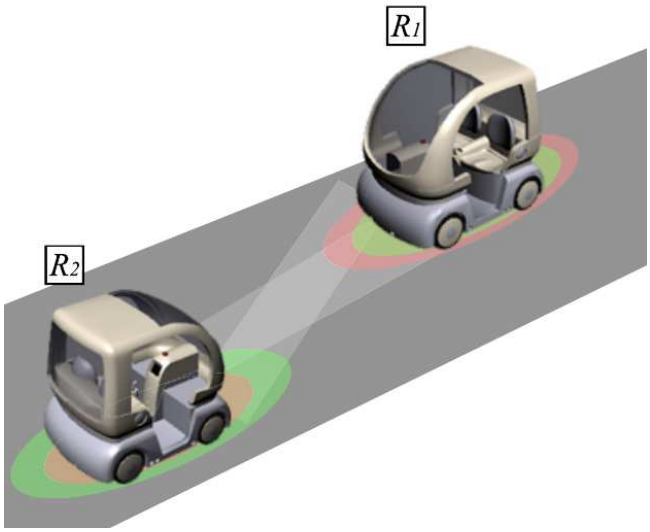


Figure 6.6: *Rumour* notion: the first robot R_1 localizes itself and sends its estimated pose to robot R_2. This one sends in turn the received poses of all the robots (only R_1 here) to all the robots in the neighborhood (only R_1 here). So R_1 can combine this information with its own pose. However since it is the same information, an over-convergence problem appears [19]).

6.3.2 State exchange

In this case the principle is to exchange the global state between the robots.

- Each robot can store the estimated state of each other and sends these states as soon as a modification appears,
- The previous are **replaced** by the received ones
- Fusion is done as soon as a change appears but the result of the fusion is **never sent**. Then no over-convergence can occur and each robot knows the last estimated state of the others.

Figures 6.7 and 6.8 depicts the operation.

This approach has also able to deal with the fleet split. Recovering the global state will be possible as soon as a robot can receive the data from at least one robot from each sub-fleet (see fig 6.9)

6.3.3 Intersection covariance

As we saw in § 6.3.1, the *rumour* issue involves usually over-convergence of the filters. Indeed the Bayesian Filters (and so the Kalman Filters) are optimistic approaches: they suppose the measurements are uncorrelated (see §3.4).

The CIF (Covariance Intersection Filter) aims at providing a **pessimistic** estimation that suppose all measurements are correlated. This filter has been introduced first in [14].

Consider two measurements a and b we want to fuse to provide c. We define:

$$\begin{aligned} a &= \tilde{a} + \bar{a} \\ b &= \tilde{b} + \bar{b} \\ c &= \tilde{c} + \bar{c} \end{aligned}$$

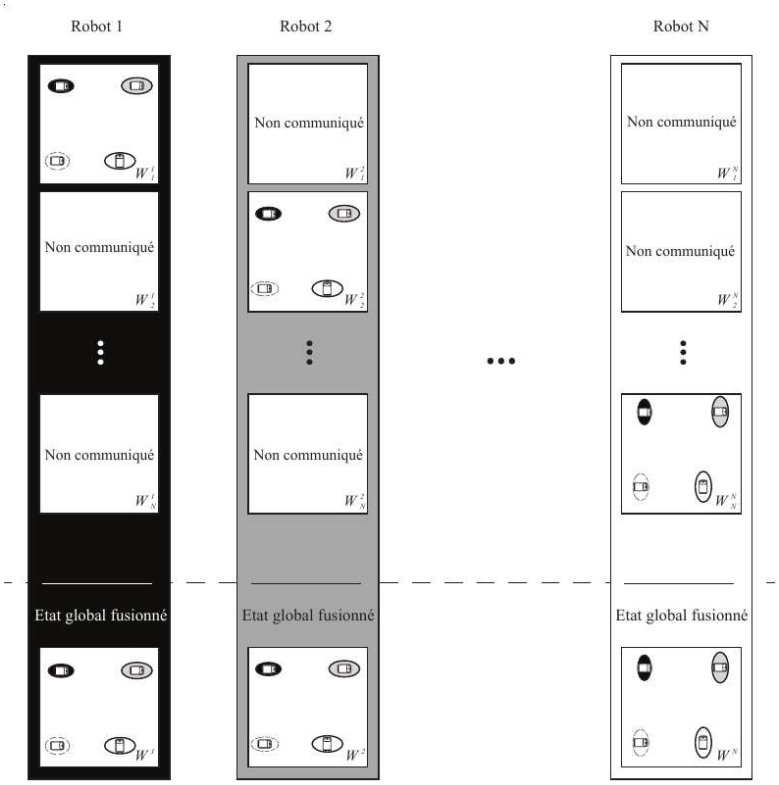


Figure 6.7: Fleet state before communication

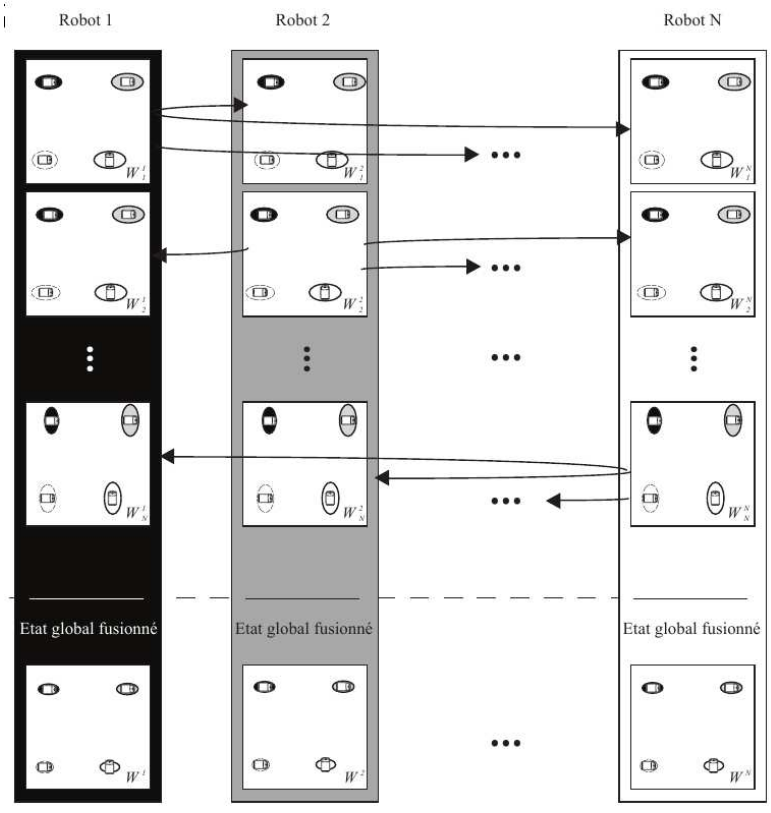


Figure 6.8: Fleet state after communication

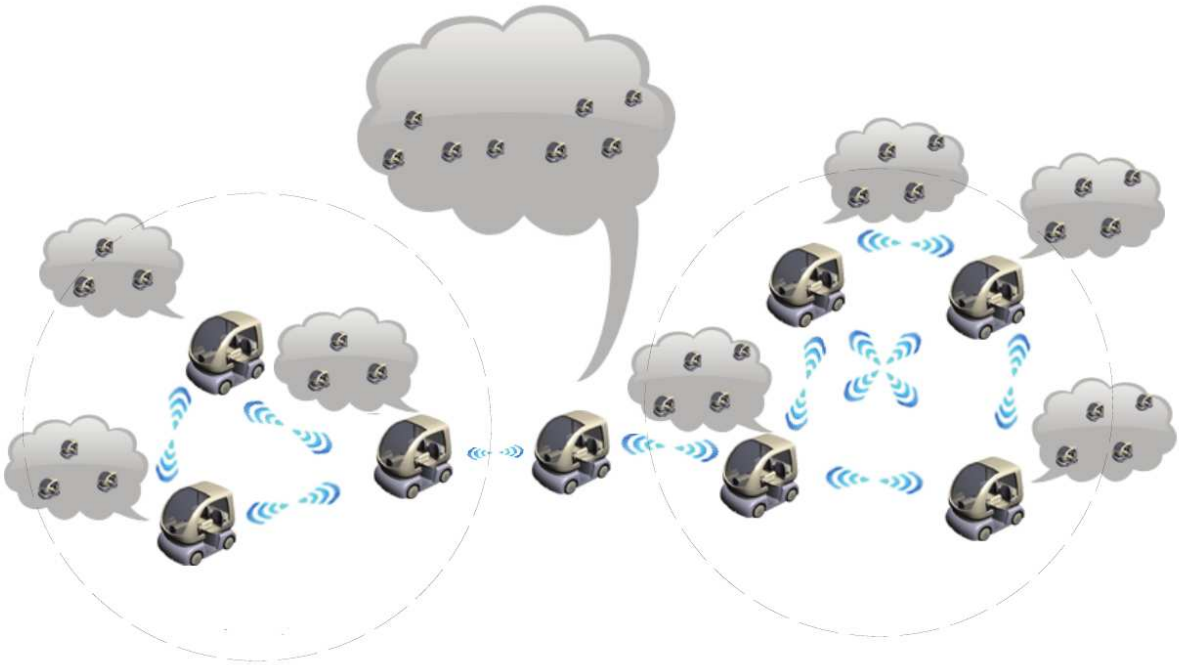


Figure 6.9: Sub-fleet diffusion using an intermediate robot

With $\bar{a}$, $\bar{b}$, $\bar{c}$ the expected values of a, b, c and $\tilde{a}$, $\tilde{b}$, $\tilde{c}$ their stochastic part. We'll have:

$$\begin{aligned}
\bar{\mathbf{P}}_{aa} &= \mathbf{Cov}[a] = \mathbf{E}\left[\tilde{a}.\tilde{a}^\top\right] \\
\bar{\mathbf{P}}_{bb} &= \mathbf{Cov}[b] = \mathbf{E}\left[\tilde{b}.\tilde{b}^\top\right] \\
\bar{\mathbf{P}}_{cc} &= \mathbf{Cov}[c] = \mathbf{E}\left[\tilde{c}.\tilde{c}^\top\right] \\
\bar{\mathbf{P}}_{ab} &= \mathbf{E}\left[\tilde{a}.\tilde{b}^\top\right]
\end{aligned}$$

The solution of Covariance Intersection Filter is (see [14]):

$$\boxed{\begin{cases} \mathbf{P}_{cc}^{-1} &= \omega\mathbf{P}_{aa}^{-1} + (1-\omega)\mathbf{P}_{bb}^{-1} \\ \bar{c} &= \mathbf{P}_{cc}\left[\omega\mathbf{P}_{aa}^{-1}\bar{a} + (1-\omega)\mathbf{P}_{bb}^{-1}\bar{b}\right] \end{cases}} \tag{6.2}$$

ω parameter is such as $\omega \in [0,1]$. Its value can be chosen to minimize either the trace or the determinant of $\mathbf{P}_{cc}$. An analysis can be found in [31, 8].

Figure 6.10 shows the behaviour of this parameter on the fusion result. We can see the result ellipse well includes the intersection of those of the two measurements.

Remarks: since the CIF is a pessimistic filter, it is sub-optimal: we cannot expect a reduction of the uncertainty after having several measurements with the same covariance. To address this issue the SCIF filter has been set up: the principle is to split the a and b covariance in two parts: 1) dependent (correlated part) and 2) independent (un-correlated part). For more details, see [25].

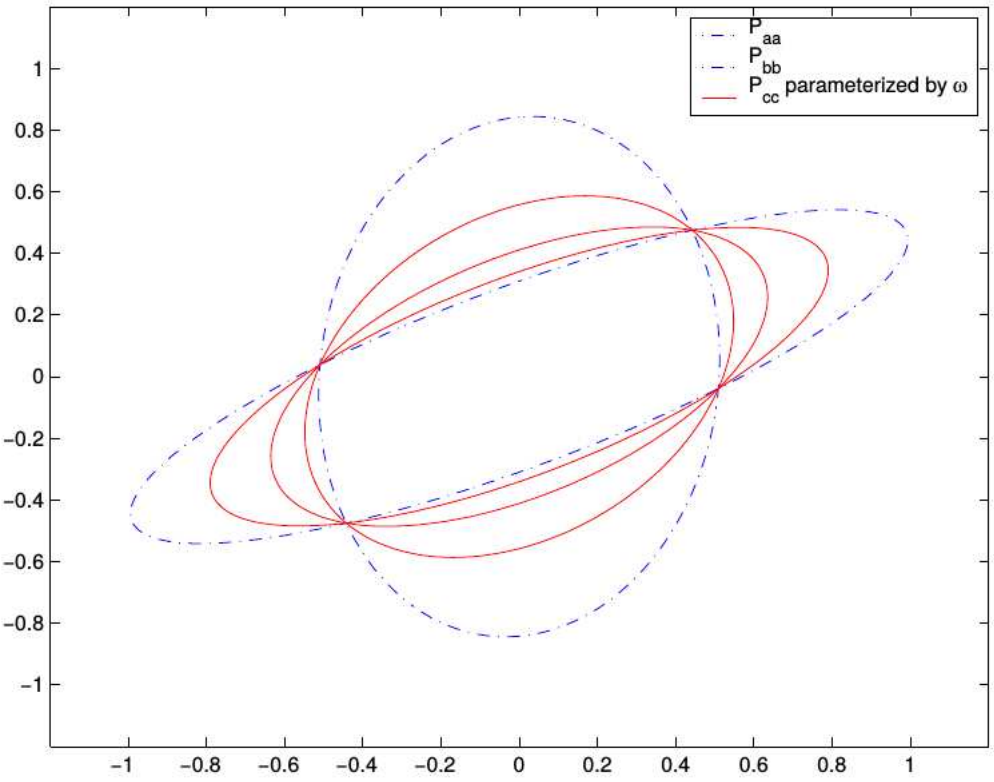


Figure 6.10: Fusion result v.s ω parameter (from [9])

Chapter 7

Data Fusion and Graphical Models

7.1 Introduction

Sometimes, a graphical representation can be worth to well represent all the actors of the problem and their dependencies. It is especially the case for SLAM where the global state can be composed of thousands of poses and landmarks (see for example 7.1 from [11]).

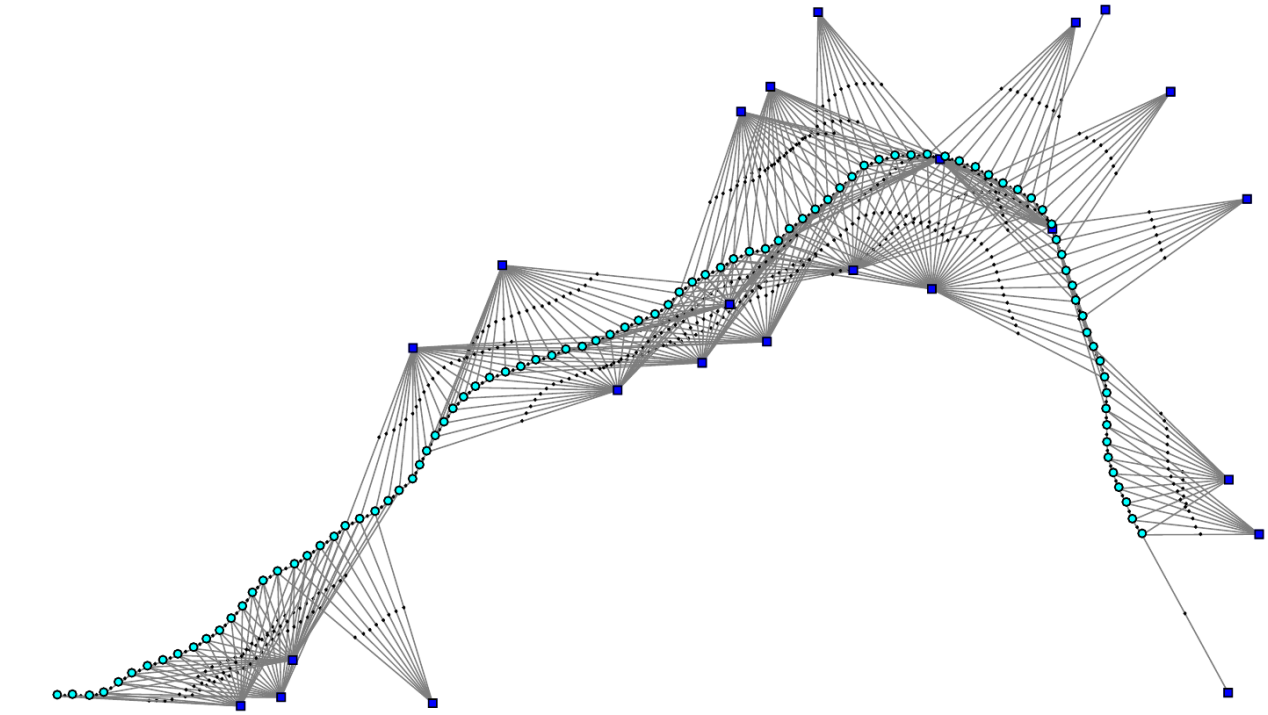


Figure 7.1: SLAM example: the problem is to estimate many landmarks position and robot poses [11]

This chapter introduces these representations and how we can deal with data fusion having such networks.

7.2 Bayes Network

7.2.1 Introduction

Bayes Network as been developed by Pearl (a comprehensive description can be found in [27]) and adapted later for dynamic systems [28].Such representations can deal with discrete and continuous

events as well.

Suppose, as an example, the following *decision* problem: a vehicle embeds both a camera and a radar to detect other vehicles ahead. Both camera and radar provides the following binary detection: a vehicle ahead is on our lane on not.

Let's name $X_k = \{0, 1\}$ the binary event "the vehicle ahead is in our lane" and R_k, C_k the detections both radar and camera has done. The question is what is the probability a vehicle ahead is really in our lane ?

We can model this problem with a **Bayes Network** as show in figure 7.2. A Bayes Network is an acyclic graph: nodes represent events and the oriented links represents the joint probability.

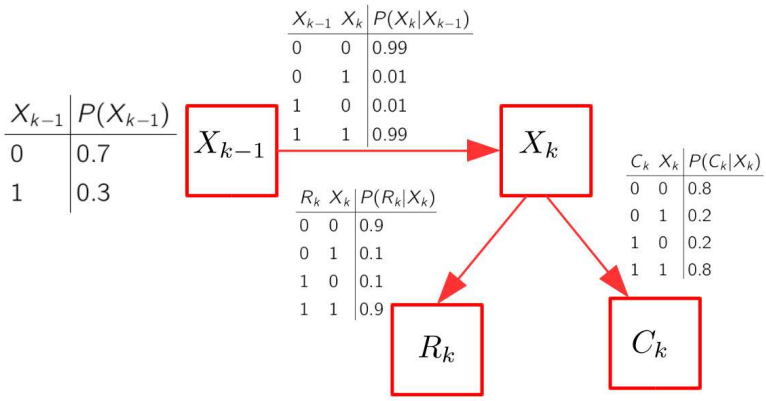


Figure 7.2: Bayes Network example, with associated joint probabilities

Suppose we got R_k detection but no C_k detection, what is $P(X_k|C_k = 0, R_k = 1)$?

This an *inference* problem. We first find the whole *global joint probability* $P(X_{k-1}, X_k, C_k, R_k)$ then we deduce (thanks to the Bayes rule) $P(X_k|C_k = 0, R_k = 1)$.

We can write the **Ancestral rule**: For X_i events ($i \in [1, n]$) we have:

$$\boxed{P(X_1, X_2, \cdots, X_n) = \prod_{i=1}^{n} P(X_i|par(X_i))} \tag{7.1}$$

Here $par(X_i)$ are the X_i parents. In our case, we'll get:

$$P(X_{k-1}, X_k, C_k, R_k) = P(X_{k-1}).P(X_k|X_{k-1}).P(C_k|X_k).P(R_k|X_k)$$

Then, we get the **marginal probability of** $P(X_k, C_k = 0, R_k = 0)$:

$$\begin{aligned} P(X_k, C_k = 0, R_k = 0) &= \sum_{X_{k-1}} P(X_{k-1}, X_k, C_k = 0, R_k = 1) \\ &= P(X_{k-1} = 0, X_k, C_k = 0, R_k = 1) + P(X_{k-1} = 1, X_k, C_k = 0, R_k = 1) \end{aligned}$$

And finally using to the Bayes Rule yields:

$$P(X_k|C_k, R_k) = \frac{P(X_k, C_k, R_k)}{P(C_k, R_k)}$$

And so :

$$P(X_k|C_k = 0, R_k = 1) = \frac{P(X_k, C_k = 0, R_k = 1)}{P(C_k = 0, R_k = 1)}$$

The reader can verify we obtain $P(X_k = 1|C_k = 0, R_k = 1) \approx 87\%$

As we saw, we can both **infer** the probabilities we want and have a **visual representation** of the problem. This is the basis of the graphical models. We'll develop now this concept for global estimation problem we usually encounter in SLAM applications.

7.2.2 Continuous Bayes Network

As we are mainly concerned by state estimation we'll try to use the same principle for continuous events (continuous random variables). In this case we'll use the *pdf* instead of probabilities. As an example consider figure 7.3 (from [11]) which represent a very simple SLAM problem (named *Toy-SLAM*). Here x_k denotes state value (pose) for time k, l_i denotes the landmarks and z_j denotes the measurements we get using the landmarks with poses x_k.

Let's define $X = (x_1, x_2, x_3, l_1, l_2)$ and $Z = (z_1, z_2, z_3, z_4)$. We group together x_k and l_i since they represent unknown of our problem.

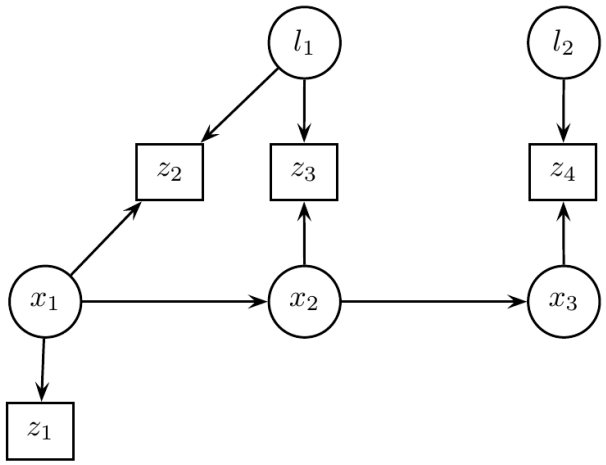


Figure 7.3: Toy-SLAM example (from[11]

Using the *ancestral rule* we can write:

$$
\begin{aligned}
p(X,Z) &= p(x_1).p(x_2|x_1).p(x_3|x_2) && (7.2)\\
&\times\ p(l_1).p(l_2) && (7.3)\\
&\times\ p(z_1|x_1) && (7.4)\\
&\times\ p(z_2|x_1,l_1).p(z_3|x_2,l_1).p(z_4|x_3,l_2) && (7.5)\\
& && (7.6)
\end{aligned}
$$

Let's detail these equations:

- eq. (7.2) is the **Markov chain** linking the pose states x_k,
- eq. (7.3) is the prior *pdf* on landmarks l_i. We can put here the knowledge we have on landmarks position. However usually in the classical SLAM process we have no such priors,
- eq. (7.4) refers to the link between the first measurement z_1 and x_1,
- eq (7.5) represents the relationships between measurements on the landmarks l_i from poses x_k.

We can write eq (7.2) to eq. (7.9) as :

$$p(X,Z) = p(Z|X).p(X) \qquad \text{with:} \quad \begin{cases} p(X) &= p(x_1).p(x_2|x_1).p(x_3|x_2) \times p(l_1).p(l_2) \\ p(X|Z) &= p(z_1|x_1) \times p(z_2|x_1,l_1).p(z_3|x_2,l_1).p(z_4|x_3,l_2) \end{cases} \tag{7.7}$$

Now, we can write the classical Bayes Rule:

$$p(X|Z) = \frac{p(X,Z)}{P(Z)} = \frac{p(Z|X)p(X)}{P(Z)} \tag{7.8}$$

As we already seen in § 2.4, it is necessary to consider X as an unknown and Z as known data and so we'll use the **likelihood function** $p(X;Z|X)$ rather than the *pdf* $p(Z|X)$. This yields:

$$p(X|Z) = \frac{p(X;Z|X)p(X)}{P(Z)} \tag{7.9}$$

Some authors denotes this likelihood function as $l(X;Z)$ to point out that X is the unknown and Z the known measurement. Moreover since $P(Z)$ is not a function to X, we can write :

$$\boxed{p(X|Z) \propto l(X;Z)p(X) \qquad \text{with} \quad l(X;Z) \triangleq p(X;Z|X)} \tag{7.10}$$

7.2.3 MAP estimation

Equation (7.10) provides the *pdf* of both pose states and landmarks. Most of the time it is necessary to deduce from $p(X|Z)$ an estimation $\hat{X}$ of X. A convenient (and classical) estimator (see § 2.5.1) is the **MAP** that can be written here as:

$$\hat{X}_{MAP} = \arg\max_X p(X|Z) = \arg\max_X \{l(X;Z).p(X)\}$$

7.3 Factor graphs

7.3.1 From Bayes Networks to Factor Graphs

Since we have $p(X|Z) = \frac{p(X,Z)}{p(Z)}$ we have therefore $p(X|Z) \propto p(X,Z)$.We can therefore rewrite eq. (7.10) as:

$$\begin{aligned} p(X|Z) \quad \propto \quad & p(x_1).p(x_2|x_1).p(x_3|x_2) \\ \times \quad & p(l_1).p(l_2) \\ \times \quad & l(x_1;z_1) \\ \times \quad & l(x_1,l_1;z_2).l(x_2,l_1;z_3).l(x_3,l_2;z_4) \end{aligned} \tag{7.11}$$

We have a set of *factors* and in order to make the factorization more clear, we use the **factor graph** presented in figure 7.4.

Since the measurements are known, they are not represented here. The 9 big black dots represent the 9 **factors** of eq. (7.13) linking the **nodes** which represent the state and landmarks unknown.

If we denote $\phi_k(x_i,x_j)$ the **factor** graph k between nodes x_i and x_j we can define the global factor graph $\phi(X)$ as:

$$\phi(X) = \prod_i \phi_i(X_i) \tag{7.12}$$

Where X_i are the set of nodes related to the factor ϕ_i. Hence we can define the global factor $\phi(l_1,l_2,x_1,x_2,x_3)$ for the toy-SLAM example as:

$$\begin{aligned} \phi(l_1,l_2,x_1,x_2,x_3) \quad = \quad & \phi_1(x_1).\phi_2(x_2,x_1).\phi_3(x_3,x_2) \\ \times \quad & \phi_4(l_1).\phi_5(l_2) \\ \times \quad & \phi_6(x_1) \\ \times \quad & \phi_7(x_1,l_1).\phi_8(x_2,l_1).\phi_9(x_3,l_2) \end{aligned} \tag{7.13}$$

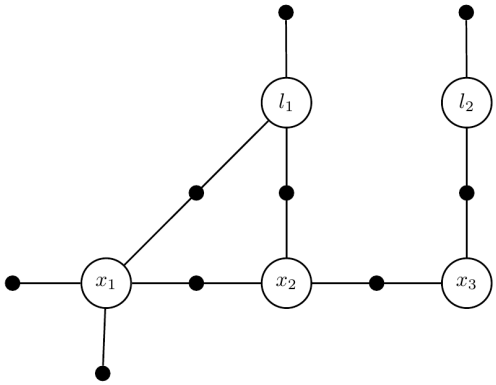


Figure 7.4: Factor graph of the toy-SLAM

7.3.2 Inference using Factor graphs

Having the global factor $\phi(X)$ we usually look for the estimation $\hat{X}$ of X. Taking the MAP yields:

$$\hat{X}_{MAP} = \arg\max_X \phi(X) = \arg\max_X \prod_i \phi_i(X_i) \tag{7.14}$$

Suppose all factors ϕ_i are of following Gaussian form (using the Mahalanobis distance, see § 8.2.4):

$$\phi_i(X_i) \propto \exp\left\{-\frac{1}{2}||h(X_i) - z_i||^2_{C_i}\right\}$$

Indeed this Gaussian shape only requires the noise is Gaussian and additive. In this case, since we can drop the -1/2 factor and taking the log of the product will lead to the minimization of the following sum:

$$\hat{X}_{MAP} = \arg\max_X \phi(X) = \arg\min_X \sum_i \left\{||h(X_i) - z_i||^2_{C_i}\right\} \tag{7.15}$$

We therefore solve our global problem by usual numerical minimization.

7.4 Remarks

- Eq. (7.15) provide an easy way to solve global fusion problems such as SLAM or others,
- Most of the time factor graphs functions are nonlinear, the minimization of eq. (7.15) requires optimization descent algorithms like Gauss-Newton or Levenberg-Marquardt techniques,
- The optimization for visual-SLAM needs to deal with 3D rotations. These specific non linear functions require to use nonlinear manifolds (see for instance[32, 11]),
- Solving the minimization of eq (7.15) without care can lead to huge computational costs, especially in the large SLAM applications. However, in SLAM applications, the **sparsity** of the factor graph involves sparse Jacobian matrices in the minimization and lead to very efficient optimizations (see for example the g^2o library [23]).

Chapter 8

Appendix

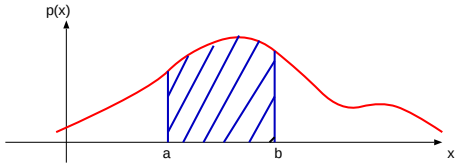
8.1 Random Values

8.1.1 Definition

A random variable x has an unknown value. We can notice:

- A Random Value **x** is characterized by its *Probability Density Function* (*pdf*) $p(x)$,
- The probability $P(x \in [a,b])$ for x to belong to $[a,b]$ will be given by:

$$P(x \in [a,b]) = \int_a^b p(x)dx$$



- Every *pdf* satisfies the normalization property:

$$\int_{-\infty}^{+\infty} p(x)dx = 1$$

Many *pdf* exist, the most often used are: **Uniform**, **Gaussian** and **Set of Dirac** (for discrete random values).

8.1.2 Properties of the random variables

For a given random variable **x** we define:

The expected value :

$$\mathbf{E}[x] = \mu_x = \int_{-\infty}^{+\infty} xp(x)dx$$

The variance :

$$\mathbf{Var}[x] = \sigma_x^2 = \mathbf{E}\left[(x-\mu_x)^2\right] = \int_{-\infty}^{+\infty} (x-\mu_x)^2 p(x)dx$$

The standard deviation :

$$\sigma_x = \sqrt{\sigma_x^2}$$

Given a constant α and two random variables x and y with $\mu_x = \mathbf{E}[x]$ and $\mu_y = \mathbf{E}[y]$, $\mathbf{Var}[x] = \sigma_x^2$ and $Var[y] = \sigma_y^2$, we'll have:

- $\mathbf{E}[x+y] = \mathbf{E}[x] + \mathbf{E}[y] = \mu_x + \mu_y$
- $\mathbf{E}[\alpha x] = \alpha \mathbf{E}[x]$
- $\mathbf{Var}[x] = \mathbf{E}\left[(x-\mu_x)^2\right] = \mathbf{E}\left[x^2\right] - \mathbf{E}[x]^2$
- $\mathbf{Var}[\alpha x] = \alpha^2 Var[x]$
- $\mathbf{Var}[x+y] = \mathbf{Var}[x] + \mathbf{Var}[y] + 2C_{xy}$
- $\mathbf{Var}[x+y] = \mathbf{Var}[x] + \mathbf{Var}[y]$ **if** x **and** y **are independent**.

8.1.3 Multivariate random variables

Given $x = (x_1, x_2, \cdots, x_n)^\top$, we also define its *pdf* $p(x)$ which is a n-D function. We define:

- **The expected value**: $\mathbf{E}[x] = \mu_x = (\mu_{x_1}, \mu_{x_2}, \cdots, \mu_{x_n})^\top$: *mean value* of x,
- **The covariance matrix**: $\mathbf{Cov}[x] = \mathbf{C}_x = \mathbf{E}\left[(x-\mu_x)(x-\mu_x)^\top\right]$

$$\mathbf{E}\left[(x-\mu_x)(x-\mu_x)^\top\right] = \begin{pmatrix} \sigma_{x_1}^2 & \sigma_{x_1x_2} & \cdots & \sigma_{x_1x_n} \\ \sigma_{x_2x_1} & \sigma_{x_2}^2 & \cdots & \sigma_{x_2x_n} \\ \vdots & \cdots & \ddots & \vdots \\ \sigma_{x_nx_1} & \sigma_{x_nx_2} & \cdots & \sigma_{x_n}^2 \end{pmatrix}$$

  - The covariance matrix expresses not only the variances of components x_1 to x_n of x but also their *covariances*,
  - If x_1 to x_n are independent, the covariance matrix will be **diagonal**.

Moreover we have the following properties:

- $\mathbf{Cov}[x] = E\left[xx^\top\right] - \mu_x\mu_x^\top$
- The covariance matrix of $\underline{Y}$ such as $\underline{Y} = \mathbf{H}x$ having $\mathbf{C}_X$: covariance matrix of x and $\mathbf{H}$: constant matrix will be given by :

$$\mathbf{Cov}[\underline{Y}] = \mathbf{H}\mathbf{C}_X\mathbf{H}^\top$$

8.2 Gaussian functions

8.2.1 One dimension Gaussian Functions

For one dimension, the Gaussian expression is

$$p(x) = \frac{1}{\sqrt{2\pi}.\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

This law is commonly used because of its properties:

- *central limit* theorem,
- a linear transformation of Gaussian *pdf* remains a Gaussian law.

We write $x \sim \mathcal{N}(\mu, \sigma^2)$. Fig 8.1 gives an example of such a Gaussian law with $\mu = 2$ and $\sigma^2 = 1$.

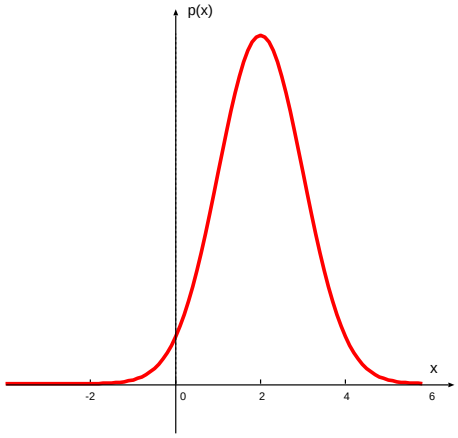


Figure 8.1: Gaussian function example: $\mathcal{N}(\mu, \sigma^2) = \mathcal{N}(2, 1)$

8.2.2 Multivariate Gaussian functions

In the Gaussian case, the n-D *pdf* is:

$$p(x) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{C}|}} \exp\left(-\frac{1}{2}(x-\mu)^\top \mathbf{C}^{-1}(x-\mu)\right)$$

With:

$$\mu = \mathbf{E}[x] \qquad \text{and} \qquad \mathbf{C} = \mathbf{Cov}[x]$$

Figure 8.2 gives a 2-D Gaussian example : $x \sim \mathcal{N}(\mu_x, \mathbf{C}_x)$ with :

$$\mu_x = (0,0)^\top \qquad \text{and} \qquad \mathbf{C}_x = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$$

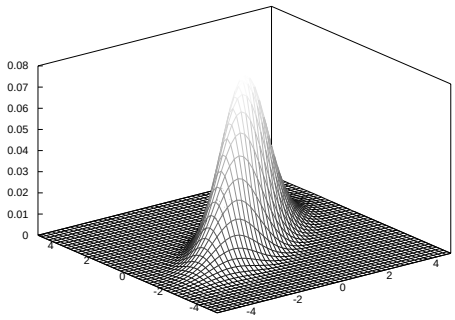


Figure 8.2: 2-D Gaussian *pdf*

8.2.3 Gaussian representation

Suppose $x = (x, y)^\top \sim \mathcal{N}(\mu_X, \mathbf{C}_X)$ with $\mu_X = (\mu_x, \mu_y)^\top$. We represent it by an ellipsis corresponding to the Gaussian function cut at a certain height (see fig 8.3).

8.2.4 Mahalanobis distance

The **Mahalanobis distance** m_{PX} between the center μ_X and a given point $\underline{P}$ is constant along this ellipsis:

$$m_{PX} = (\underline{P} - \underline{\mu}_X)^\top \mathbf{C}^{-1} (\underline{P} - \underline{\mu}_X)$$

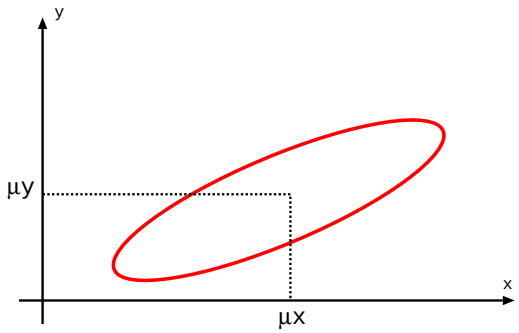


Figure 8.3: Ellipse representation of a Gaussian function

Sometimes the following notation is used:

$$m_{PX} \stackrel{\Delta}{=} ||\underline{P} - \underline{\mu}_X||^2_{\mathbf{C}} = (\underline{P} - \underline{\mu}_X)^\top \mathbf{C}^{-1} (\underline{P} - \underline{\mu}_X) \tag{8.1}$$

And, more generally, a multivariate Gaussian function can be written with this notation:

$$\boxed{\mathcal{N}(x; \mu_x, \mathbf{C}_x) \propto \exp\left(-\frac{1}{2}||x - \mu_x||^2_{\mathbf{C}_x}\right)} \tag{8.2}$$

8.2.5 Canonical Gaussian parameterization

A Gaussian function with **Moments parametrization** $\mathcal{N}(\mu, \mathbf{C})$ has a **canonical representation** [28]:

Moments to canonical parameterization

$$\mathcal{N}(x; \mu, \mathbf{C}) = \phi(x; g, h, \mathbf{K}) = \exp\left(g + x^\top h - \frac{1}{2} x^\top \mathbf{K} x\right)$$

$$\text{with} \quad \begin{cases} \mathbf{K} &= \mathbf{C}^{-1} \\ h &= \mathbf{C}^{-1}\mu \\ g &= -\frac{1}{2}\log\left[(2\pi)^n |\mathbf{C}|\right] - \frac{1}{2}\mu^\top \mathbf{C}^{-1}\mu = Cte \end{cases}$$

Canonical to moments parameterization

$$\phi(x;\ g, h, \mathbf{K}) = k\mathcal{N}(x; \mu, \mathbf{C})$$

$$\text{with:} \quad \begin{cases} \mathbf{C} &= \mathbf{K}^{-1} \\ \underline{\mu} &= \mathbf{K}^{-1} h \\ k &= (2\pi)^{n/2} |\mathbf{K}|^{-1/2} \exp\left(g + \frac{1}{2} h^\top \mathbf{K}^{-1} h\right) \end{cases}$$

Operations on canonical representation

This representation simplifies operations on Gaussian functions:

$$\phi(g_1, h_1, \mathbf{K}_1) \times \phi(g_2, h_2, \mathbf{K}_2) = \phi(g_1 + g_2, h_1 + h_2, \mathbf{K}_1 + \mathbf{K}_2)$$

$$\frac{\phi(g_1, h_1, \mathbf{K}_1)}{\phi(g_2, h_2, \mathbf{K}_2)} = \phi(g_1 - g_2, h_1 - h_2, \mathbf{K}_1 - \mathbf{K}_2)$$

8.2.6 Gaussian product

The product of two Gaussians $\mathcal{N}(\mu_1, \mathbf{C}_1)$ and $\mathcal{N}(\mu_2, \mathbf{C}_2)$ will be given by:

$$\phi(g_1, h_1, \mathbf{K}_1) \times \phi(g_2, h_2, \mathbf{K}_2) = \phi(g_1 + g_2, h_1 + h_2, \mathbf{K}_1 + \mathbf{K}_2)$$

With

$$\mathbf{K}_1 = \mathbf{C}_1^{-1} \text{ , } h_1 = \mathbf{C}_1^{-1}\mu_1 \text{ , } \mathbf{K}_2 = \mathbf{C}_2^{-1} \text{ and } h_2 = \mathbf{C}_2^{-1}\mu_2$$

The product of two Gaussians $\mathcal{N}(\mu_1, \mathbf{C}_1)$ and $\mathcal{N}(\mu_2, \mathbf{C}_2)$ will be a Gaussian function (un-normalized) given by:

$$\mathcal{N}(\mu_1, \mathbf{C}_1) \times \mathcal{N}(\mu_2, \mathbf{C}_2) = k\mathcal{N}(\mu, \mathbf{C})$$

With:

$$\mathbf{C} = \left[\mathbf{C}_1^{-1} + \mathbf{C}_2^{-1}\right]^{-1}$$

$$^{-} = \left[\mathbf{C}_1^{-1} + \mathbf{C}_2^{-1}\right]^{-1}\left[\mathbf{C}_1^{-1}\mu_1 + \mathbf{C}_2^{-1}\mu_2\right]$$

8.3 Stochastic Linear systems

8.3.1 Continuous to discrete stochastic state systems

Consider the following linear stochastic system:

$$\left\{\begin{array}{rcl} \dot{x}(t) & = & \mathbf{A}_c x(t) + \mathbf{B}_c u(t) + \mathbf{M}_c w(t) \\ y(t) & = & \mathbf{C}x(t) + v(t) \end{array}\right.$$

We can transform this continuous system in a discrete one. T_s is the *sampling time* between times $k-1$ and k.

$$\left\{\begin{array}{rcl} x_k & = & \mathbf{A}x_{k-1} + \mathbf{B}u_k + w_k \\ y_k & = & \mathbf{C}x_k + v_k \end{array}\right. \quad \text{with:} \quad \left\{\begin{array}{rcll} \mathbf{A} & = & \exp(\mathbf{A}_c T_s) & \\ \mathbf{B} & = & \left[\int_0^{T_s} \exp(\mathbf{A}_c.(T_s - t))dt\right]\mathbf{B}_c & \\ & = & \mathbf{A}_c^{-1}\left[\mathbf{A} - \mathbf{I}\right]\mathbf{B}_c & \text{if } \mathbf{A}_c \text{ is invertible} \\ \mathbf{Q}_k & = & \int_0^{T_s} \exp(\mathbf{A}_c t)\mathbf{M}_c\mathbf{Q}_c\mathbf{M}_c^{\top}\exp(\mathbf{A}_c^{\top} t)dt & \end{array}\right.$$

Where $\mathbf{Q}_c$ is the covariance matrix of white noise $w(t)$

8.3.2 Movement Modeling

Sometimes, we don't know exactly how to model the systems, the physical phenomenon is either unknown, or too complicated. As examples, we can have pedestrian movements, targets evolution (military applications), etc. So, we consider mainly the basic modellings:

- Static evolution,
- Constant velocity,
- Constant acceleration.

Static evolution

We assume the state vector $x(t)$ is approximately constant but with a noise $w(t)$ with variance σ_w^2 on its evolution: We have:

$$\dot{x}(t) = w(t)$$

In this case matrices $\mathbf{A}_c$ and $\mathbf{B}_c$ are null and we obtain $\mathbf{A} = \mathbf{I}$, $\mathbf{B} = 0$. We get:

$$\boxed{x_k = x_{k-1} + w_k \qquad \text{with} \quad \mathbf{Var}[w_k] = T_s \sigma_w^2}$$

Constant velocity

Here we assume $X = (x, \dot{x})^\top$ and a constant speed, the acceleration is therefore null with noise $w(t)$ having a variance σ_w^2:

$$\dot{X}(t) = \begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}}_{\mathrm{A}_c} \begin{pmatrix} x \\ \dot{x} \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\mathrm{M}_c} w(t)$$

$\mathbf{A}_c$ cannot be inverted, we need to solve the integral. We note that:

$$\mathbf{A} = \exp(\mathbf{A}_c T_s) = \mathbf{I} + \mathbf{A}_c T_s + \frac{(\mathbf{A}_c T_s)^2}{2!} + \cdots$$

And:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & T_s \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & T_s \\ 0 & 1 \end{pmatrix}$$

Finally we have:

$$\boxed{X_k = \begin{pmatrix} x_k \\ \dot{x}_k \end{pmatrix} = \begin{pmatrix} 1 & T_s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{pmatrix} + w_k = \mathbf{A} x_{k-1} + w_k}$$

The variance $\mathbf{Q}$ of w_k is given by:

$$\mathbf{Q}_k = \int_0^{T_s} \exp(\mathbf{A}_c t) \mathbf{M}_c \mathbf{Q}_c \mathbf{M}_c^\top \exp(\mathbf{A}_c^\top t) dt$$

$$\mathbf{Q}_k = \int_0^{T_s} \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \sigma_w^2 \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t & 1 \end{pmatrix} dt$$

So:

$$\mathbf{Cov}[w_k] = \mathbf{Q}_k = \sigma_w^2 \begin{pmatrix} \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^2}{2} & T_s \end{pmatrix}$$

Constant Acceleration

Here, we assume $X = (x, \dot{x}, \ddot{x})^\top$. We have:

$$\dot{X}(t) = \begin{pmatrix} \dot{X} \\ \ddot{x} \\ \dddot{x} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w(t)$$

We obtain the following model:

$$\boxed{X_k = \begin{pmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \end{pmatrix} = \begin{pmatrix} 1 & T_s & \frac{T_s^2}{2} \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ \dot{x}_{k-1} \\ \ddot{x}_{k-1} \end{pmatrix} + w_k = \mathbf{A} x_{k-1} + w_k}$$

With:

$$\mathbf{Q}_k = \mathbf{Cov}\left[w_k\right] = \sigma_w^2 \begin{pmatrix} \frac{T_s^5}{20} & \frac{T_s^4}{8} & \frac{T_s^3}{6} \\ \frac{T_s^4}{8} & \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^3}{6} & \frac{T_s^2}{2} & T_s \end{pmatrix}$$

Bibliography

[1] Iyad Abuhadrous. *Système embarqué temps réel de localisation et de modélisation 3D par fusion multi-capteur*. PhD thesis, Ecole des Mines de Paris, Paris, Janvier 2005.

[2] D.L. Alspach and H.W. Sorenson. Non-linear bayesian estimation using gaussian sum approximation. *IEEE Transaction on Automatica Control*, 17:439–447, 1972.

[3] Brian. D. O. Anderson and John. B. Moore. *Electrical Engineering, Optimal Filtering*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, USA, 1979.

[4] M.S. Arulampalam, S. Maskel, N. Gordon, and T.Clapp. A tutorial on particles filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transaction on Signal Processing*, 50(2):174–188, 2002.

[5] Romuald Aufrère, Nadir Karam, Frédéric Chausse, and Roland Chapuis. A state exchange approach in real conditions for multi-robot cooperative localization. In *International Conference on Intelligent Robots andSystems IROS*, Taipei, Taiwan, 18–22 Oct. 2010.

[6] Yaakov Bar-Shalom, Peter K Willett, and Xin Tian. *Tracking and data fusion*. YBS publishing, 2011.

[7] Federico Castanedo. A review of data fusion techniques. *The Scientific World Journal*, 2013, 2013.

[8] Lingji Chen, Pablo O Arambel, and Raman K Mehra. Estimation under unknown correlation: covariance intersection revisited. *IEEE Transactions on Automatic Control*, 47(11):1879–1882, 2002.

[9] Lingji Chen, Pablo O Arambel, and Raman K Mehra. Fusion under unknown correlation-covariance intersection as a special case. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 2, pages 905–912. IEEE, 2002.

[10] D. Crisan and A. Doucet. Survey of convergence results on particle filtering methods for practitioners. *IEEE Transaction on Signal Processing*, 50(3):736–746, 2002.

[11] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[12] A. Gelb. *Applied Optimal estimation*. MIT press, Cambridge Mass, 1974.

[13] Simon J. Julier and Jeffrey K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *In Proceedings of the American Control Conference*, pages 2369–2373, 1997.

[14] Simon J Julier and Jeffrey K Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *American Control Conference, 1997. Proceedings of the 1997*, volume 4, pages 2369–2373. IEEE, 1997.

[15] SJ Julier and Jeffrey K Uhlmann. General decentralized data fusion with covariance intersection. *Handbook of multisensor data fusion: theory and practice*, pages 319–344, 2009.

[16] S.J. Julier and J.K. Ulhmann. Unscented filtering and nonlinear estimation. *IEEE Review*, 92(3), Mars 2004.

[17] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 82:34–45, 1960.

[18] R. E. Kalman and R. Bucy. A new approach to linear filtering and prediction theory. *Trans. ASME, Journal of Basic Engineering*, 83:95–108, 1961.

[19] Nadir Karam. *Agrégation de données décentralisées pour la localisation multi-véhicules*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II – France, 2009.

[20] Nadir Karam, Frédéric Chausse, Romuald Aufrère, and Roland Chapuis. Collective localization of communicant vehicles applied to collision avoidance. In *9th International IEEE Conference on Intelligent Transportation Systems*, Toronto, Canada, September 2006.

[21] Nadir Karam, Frédéric Chausse, Romuald Aufrère, and Roland Chapuis. Localization of a group of communicating vehicles by state exchange. In *IEEE International Conference on Intelligent Robots and Systems (IROS06)*, Beijing China, October 2006.

[22] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[23] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[24] Laetitia Lamard, Roland Chapuis, and Jean-Philippe Boyer. Multi target tracking with cphd filter based on asynchronous sensors. In *International Conference on Information Fusion, Istanbul*, July 2013.

[25] Hao Li. *Cooperative perception: Application in the context of outdoor intelligent vehicle systems*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 2012.

[26] P.S. Maybeck. *Stochastics models, estimation and control*. Academic Press, New York, USA, 1979.

[27] Kevin Murphy. A brief introduction to graphical models and bayesian networks. 1998.

[28] Kevin Patrick Murphy. Dynamic bayesian networks: Representation, inference and learning, 2002.

[29] Esha D. Nerurkar, Stergios I. Roumeliotis, and Agostino Martinelli. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In *Proceedings, 2009 IEEE International Conference on Robotics and Automation*, pages On DVD–ROM, 2009.

[30] A. Papoulis. Maximum entropy and spectral estimation: A review. *j-ieee-assp*, 29, 1981.

[31] Marc Reinhardt, Benjamin Noack, and Uwe D Hanebeck. Closed-form optimization of covariance intersection for low-dimensional matrices. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 1891–1896. IEEE, 2012.

[32] Geraldo Silveira, Ezio Malis, and Patrick Rives. An efficient direct approach to visual slam. *IEEE transactions on robotics*, 24(5):969–979, 2008.

[33] H.W. Sorenson and D.L. Alspach. Recursive bayesian estimation using gaussian sums. *Automatica*, 7:465–479, 1971.

[34] Cédric Tessier, Christophe Cariou, Christophe Debain, Frédéric Chausse, Roland Chapuis, and Christophe Rousset. A real-time, multi-sensor architecture for fusion of delayed observations: application to vehicle localization. In *Proc. IEEE Intelligent Transportation Systems Conference ITSC '06*, pages 1316–1321, 17–20 Sept. 2006.