Chapter 6

Mobile Robot Localisation

One of the basic functions of mobile robots is to traverse from a certain position to another one in the environment. In order to accomplish this task the mobile robot needs to know its pose in the environment at any time to determine whether it has reached its final destination; such process is called localisation. There are many techniques to approach the localisation problem and they differ in the type of sensors used and how the uncertainty is addressed. In this chapter, three popular localisation techniques are described, and their advantages and disadvantages are illustrated.

6.1 Motion-based Localisation (Dead Reckoning)

This technique uses the internal kinematics of the robot to localise it in the environment. The robot may also employ some proprioceptive sensors such as: encoders or IMUs, to provide a better estimate of the pose change over time. This method is simple to implement, and does not require sophisticated sensors. However, such technique suffers from the growth of uncertainty about the robot pose over time due to the numerical integration and accumulation of error. This problem is illustrated in the following example: Consider a differential-drive robot that operates in a 2D environment, shown in Figure 6.1, with kinematic model given by equation (4.1), where v and ω are the linear and angular velocities of the robot, respectively. v and ω represent the robot inputs.

$$\frac{d}{dt}\begin{bmatrix} s_x \\ s_y \\ s_\theta \end{bmatrix} = \begin{bmatrix} \cos(s_\theta) & 0 \\ \sin(s_\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6.1}$$

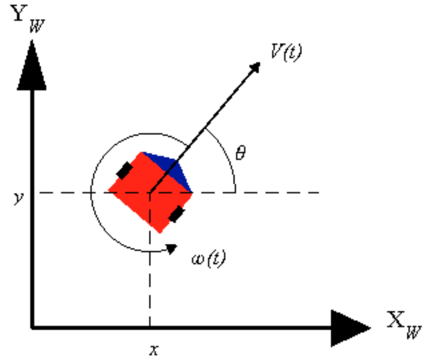


Figure 6.1: Differential-drive robot pose $\boldsymbol{s}_k = \begin{bmatrix} s_x & s_y & s_\theta \end{bmatrix}^T$. The robot inputs are the linear and angular velocities $\boldsymbol{u}_k = [v \ \omega]^T$.

If Δt is the sampling time, then it is possible to compute the incremental linear and angular displacements, Δd and $\Delta\theta$, as follows:

$$\begin{aligned} \Delta d &= v.\Delta t \\ \Delta\theta &= \omega \cdot \Delta t. \end{aligned} \tag{6.2}$$

To compute the pose of the robot at any given time step, equation (6.1) can be numerically integrated as shown in equation (6.3). This approximation follows the Markov assumption where the current robot pose depends only on the previous pose and the input velocities.

$$\begin{bmatrix} s_{x,k} \\ s_{y,k} \\ s_{\theta,k} \end{bmatrix} = \begin{bmatrix} s_{x,k-1} \\ s_{y,k-1} \\ s_{\theta,k-1} \end{bmatrix} + \begin{bmatrix} \Delta d \cos\left(s_{\theta,k-1}\right) \\ \Delta d \sin\left(s_{\theta,k-1}\right) \\ \Delta\theta \end{bmatrix} \tag{6.3}$$

If the mobile robot is equipped with encoders attached to each wheel, where they measure the wheels displacement, Δd_r and Δd_l, during each time step, the incremental linear and angular displacements are computed as shown in equation (6.4), where l is the robot wheel base.

$$\begin{bmatrix} \Delta d \\ \Delta \theta \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ 1/l & -1/l \end{bmatrix} \begin{bmatrix} \Delta d_r \\ \Delta d_l \end{bmatrix} \tag{6.4}$$

The pose estimation of a mobile robot is almost always associated with some uncertainty with respect to its state parameters. There are several factors that contribute to such uncertainty, where some are due to deterministic (systematic) errors and others are due to nondeterministic (random) errors. The deterministic errors can be identified and compensated during estimation, and they include: misalignment of wheels and unequal wheel diameter. On the other hand, nondeterministic errors, such as slipping and drifting, are random and they cause the growth of uncertainty over time. From a geometric point of view, the error in differential-drive robots is classified into three groups:

- Range error: it is associated with the computation of Δd over time.
- Turn error: it is associated with the computation of $\Delta \theta$ over time.
- Drift error: it is associated with the difference between the angular speed of the wheels and it affects the error in the angular rotation of the robot.

Due to such uncertainty, it is possible to represent the belief of the robot pose by a Gaussian distribution, where the mean vector $\boldsymbol{\mu}_k$ is the best estimate of the pose and the covariance matrix $\boldsymbol{\Sigma}_k$ is the uncertainty of the pose that encapsulates the above errors. This distribution is denoted by $\mathbf{s}_k \sim \mathcal{N}\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$.

In the context of probability, the robot pose at time step k, denoted by $\mathbf{s}_k$, can be described with Markov assumption as function of all previous robot pose $\mathbf{s}_{k-1}$ and the current control input $\mathbf{u}_k = [v_k \;\; \omega_k]^T$. This process is called the robot motion model (state transition model) and it is defined as follows:

$$\mathbf{s}_k = \mathbf{h}\left(\mathbf{s}_{k-1}, \mathbf{u}_k\right) + \mathbf{q}_k, \tag{6.5}$$

where $\mathbf{q}_k$ is an additive Gaussian noise such that $\mathbf{q}_k \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_k\right)$, and $\mathbf{Q}_k$ is a positive semidefinite covariance matrix. This noise is directly related to the error sources described above. The function $\mathbf{h}\left(\mathbf{s}_{k-1}, \mathbf{u}_k\right)$ is generally nonlinear, and in the case of a differential-drive robot illustrated in Figure 6.1, this function is defined as:

$$\mathbf{h}\left(\mathbf{s}_{k-1}, \mathbf{u}_k\right) = \begin{bmatrix} s_{x,k-1} + \Delta t.\ V_k.\cos\left(s_{\theta,k-1}\right) \\ s_{y,k-1} + \Delta t.\ V_k.\sin\left(s_{\theta,k-1}\right) \\ s_{\theta,k-1} + \Delta t.\ \omega_k \end{bmatrix}, \tag{6.6}$$

Assume that the robot pose at time step $k-1$ is given by a Gaussian distribution such that $\mathbf{s}_{k-1} \sim \mathcal{N}\left(\boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1}\right)$. Then, the above setup can be used to estimate the robot pose at time step k by linearizing the robot motion model (state transition model) using first-order Taylor expansion around $\boldsymbol{\mu}_k$ as follows:

$$\boldsymbol{\mu}_k = \mathbf{h}\left(\boldsymbol{\mu}_{k-1}, \mathbf{u}_k\right), \tag{6.7}$$

$$\mathbf{H}_k = \nabla_{\mathbf{s}_{k-1}} \mathbf{h}\left(\mathbf{s}_{k-1}, \mathbf{u}_k\right)\Big|_{\mathbf{s}_{k-1}=\boldsymbol{\mu}_{k-1}}, \tag{6.8}$$

$$\mathbf{s}_k \approx \boldsymbol{\mu}_k + \mathbf{H}_k\left(\mathbf{s}_{k-1} - \boldsymbol{\mu}_{k-1}\right). \tag{6.9}$$

Equation (6.8) represents the Jacobian matrix of $\mathbf{h}\left(\mathbf{s}_{k-1}, \mathbf{u}_k\right)$ with respect to each variable in $\mathbf{s}_{k-1}$, evaluated at $\mathbf{s}_{k-1} = \boldsymbol{\mu}_{k-1}$. In the case of a differential-drive robot, the Jacobian $\mathbf{H}_k$ is computed as follows:

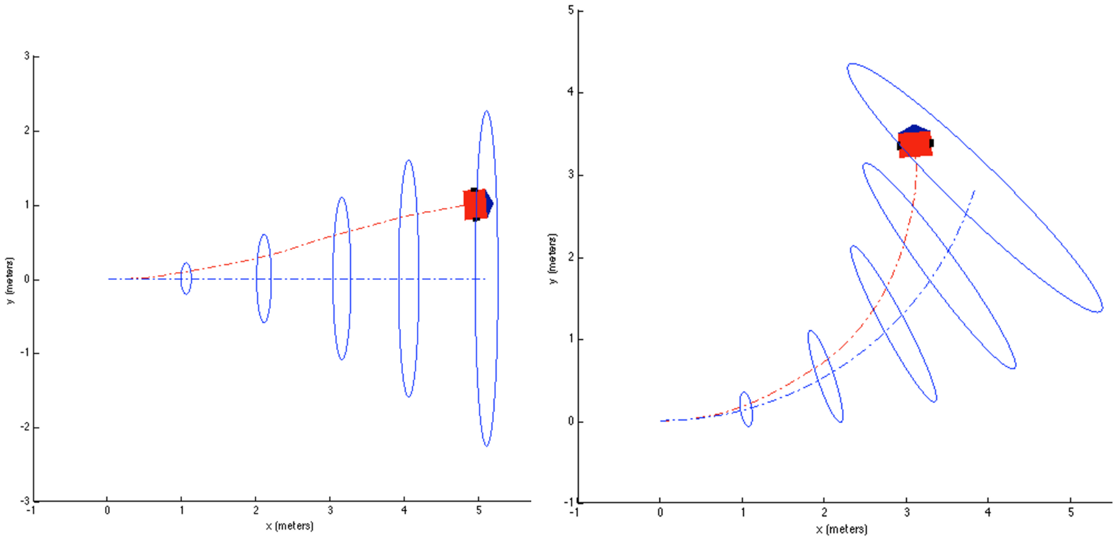
$$\mathbf{H}_k = \begin{bmatrix} \frac{\partial h_1}{\partial s_{x,k-1}} & \frac{\partial h_1}{\partial s_{y,k-1}} & \frac{\partial h_1}{\partial s_{\theta,k-1}} \\ \frac{\partial h_2}{\partial s_{x,k-1}} & \frac{\partial h_2}{\partial s_{y,k-1}} & \frac{\partial h_2}{\partial s_{\theta,k-1}} \\ \frac{\partial h_3}{\partial s_{x,k-1}} & \frac{\partial h_3}{\partial s_{y,k-1}} & \frac{\partial h_3}{\partial s_{\theta,k-1}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta t. v_k.\sin(\mu_{\theta,k-1}) \\ 0 & 1 & \Delta t.\ v_k.\cos(\mu_{\theta,k-1}) \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.10}$$

Since the robot motion model is linearised and all uncertainties are Gaussians, it is possible to compute the covariance $\boldsymbol{\Sigma}_k$ associated with the robot pose at time step k using the properties of Gaussians as follows:

$$\boldsymbol{\Sigma}_k = \mathbf{H}_k \boldsymbol{\Sigma}_{k-1} \mathbf{H}_k^T + \mathbf{Q}_k \tag{6.11}$$

Thus, the estimated pose at time step k is Gaussian such that $\mathbf{s}_k \sim \mathcal{N}\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$, and it is computed recursively using the pose at time step $k-1$ and the input vector $\mathbf{u}_k$. The initial robot pose is assumed known such that $\boldsymbol{\mu}_k = \mathbf{0}$, and $\boldsymbol{\Sigma}_k = \mathbf{0}$.

According to equation(6.11), the pose uncertainty will always increase every time the robot moves due to the addition of the nondeterministic error represented by $\mathbf{Q}_k$, which is positive semi-definite. This result is illustrated in Figure 6.2 where the joint uncertainty of s_x and s_y is represented by the ellipsoid around the robot. In Figure 6.2(a), as the robot moves along the x-axis, its uncertainty along the y-axis increases faster than the x-axis due to the drift error. Figure 6.2(b) shows that the uncertainty ellipsoid is no longer perpendicular to the motion direction as soon as the robot starts to turn.



(a)

(b)

Figure 6.2: Uncertainty growth for differential-drive robot where the blue-dashed line is the estimated pose, red-dashed line is the true pose subject to nondeterministic errors, and the blue ellipsoid represents 99% confidence of the pose in s_x and s_y. (a) Robot with straight move command only; (b) robot with straight and turn move commands.

With this type of localisation, where the robot depends solely on its proprioceptive sensors, the pose uncertainty will grow over time as the robot moves. This growth will soon make the estimated pose invalid for the robot to make proper navigational decisions, such as performing optimal path planning. One way to overcome this issue is to use a map; this approach is called map-based localisation and it is discussed in Section 6.2.

6.1.1 Pose covariance matrix

Consider the following robot motion model (state transition model):

$$\mathbf{h}\left(\mathbf{s}_k, \omega_{r,k}, \omega_{l,k}\right) = \begin{bmatrix} s_{x,k-1} + r\Delta t \frac{\omega_{r,k}+\omega_{l,k}}{2} cos\left(s_{\theta,k-1}\right) \\ s_{y,k-1} + r\Delta t \frac{\omega_{r,k}+\omega_{l,k}}{2} sin\left(s_{\theta,k-1}\right) \\ s_{\theta,k-1} + r\Delta t \frac{\omega_{r,k}-\omega_{l,k}}{l} \end{bmatrix} \tag{6.12}$$

where $\omega_{r,k}$ and $\omega_{l,k}$ are the angular velocities of the right and left wheels respectively at time step k. These values can be estimated either using encoders or motion model. Now assume the noise in both right and left wheel angular velocities to be zero-mean Gaussian distribution such that:

$$\begin{bmatrix} \omega_{r,k} \\ \omega_{l,k} \end{bmatrix} \sim \mathcal{N}\left(0, \Sigma_{\Delta,k}\right), \tag{6.13}$$

$$\Sigma_{\Delta,k} = \begin{bmatrix} k_r \left|\omega_{r,k}\right| & 0 \\ 0 & k_l \left|\omega_{l,k}\right| \end{bmatrix}, \tag{6.14}$$

where k_r and k_l are constants representing the error associated with computing the angular velocity by each wheel. These constants are related to the traction between the wheels and the floor surface or the encoder noise used to compute the wheel displacements. Note that according to (6.14) the variances of the angular velocity of right and left wheels are independent. In addition, they vary by the variation of the absolute value of the angular velocity. For instance, larger angular speed of the right motor $|\omega_{r,k}|$ will lead to a larger variance of that motor $k_r \left|\omega_{r,k}\right|$. It is possible to propagate this noise $\Sigma_{\Delta,k}$ to be seen from the robot state perspective using Taylor series expansion as follows:

$$Q_k = \nabla_{\omega_k}\mathbf{h} \cdot \Sigma_{\Delta,k} \cdot \left(\nabla_{\omega_k}\mathbf{h}\right)^T \tag{6.15}$$

$$\nabla_{\omega_k}\mathbf{h} = \begin{bmatrix} \frac{\partial h_1}{\partial \omega_{r,k}} & \frac{\partial h_1}{\partial \omega_{l,k}} \\ \frac{\partial h_2}{\partial \omega_{r,k}} & \frac{\partial h_2}{\partial \omega_{l,k}} \\ \frac{\partial h_3}{\partial \omega_{r,k}} & \frac{\partial h_3}{\partial \omega_{l,k}} \end{bmatrix} = \frac{1}{2} r \Delta t \begin{bmatrix} \cos(s_{\theta,k-1}) & \cos(s_{\theta,k-1}) \\ \sin(s_{\theta,k-1}) & \sin(s_{\theta,k-1}) \\ \frac{2}{l} & -\frac{2}{l} \end{bmatrix} \tag{6.16}$$

6.2 Map-based Localisation

In this approach, the robot utilises a map of the environment along with some exteroceptive sensors, such as LiDAR or camera, to localise itself within the map. This technique is superior to the dead-reckoning localisation because the uncertainty of the pose does not grow, but it is limited by the noise of the exteroceptive sensor. Figure 6.3 shows the major differences between dead-reckoning localisation and map-based localisation.

Dead-reckoning localisation:	**Map-based localisation:**
1. Kinematic/dynamic model	1. Kinematic/dynamic model
2. Proprioceptive sensors:	2. Proprioceptive sensors:
- Encoders	- Encoders
- Inertial measurement unit (IMU)	- Inertial measurement unit (IMU)
	3. Exteroceptive sensors:
	- LiDAR
	- Camera
	4. Map of the environment
(a)	(b)

Figure 6.3: Requirement for: (a) dead-reckoning localisation, (b) map-based localisation.

Map-based localisation requires a map that contains useful information about the environment, and it includes landmarks. These landmarks can be simple features such as: points, lines and planes, or they can be more complex features such as: textures, arbitrary shapes, etc. The type of the existing exteroceptive sensor determines the complexity of the map features. Of course, complex the feature in the map, the more processing needed prior localisation. In this course, only point and lines are considered. The landmarks are denoted by $\mathbf{m}_i$. The set of all landmarks in the environment represents the map and it is denoted by M, such that:

$$M = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{n_l}\} \tag{6.17}$$

where n_l is the total number of landmarks in the environment. There are many map-based localisation techniques, however, the most popular are the probabilistic techniques, where the robot pose is represented by probability distribution over the state space, for example $\mathbf{s}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Almost all probabilistic localisation techniques use Bayes filters, and they follow Markov assumption, where the current state of the robot, e.g. the pose, depends on the current inputs and only on the immediate previous pose. Also, the Markov assumption requires that the observation at time step k depends only on the state at that time step.

In addition to the robot motion model (state transition model) described in motion-based localisation, map-based localisation uses its exteroceptive sensor to detect landmarks in the environment and match them to landmarks in the map. This process corresponds to the robot observation model, and it is defined as follows:

$$\mathbf{z}_{i,k} = \mathbf{g}\left(\mathbf{m}_i, \mathbf{s}_k\right) + \mathbf{r}_{i,k}, \tag{6.18}$$

where $\mathbf{z}_{i,k}$ is the measurement coming from the exteroceptive sensor that correspond to the i-th landmark in the map, and $\mathbf{r}_{i,k}$ is an additive Gaussian noise such that $\mathbf{r}_{i,k} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_k\right)$ where $\mathbf{R}_k$ its positive semidefinite covariance matrix. The function $\mathbf{g}\left(\mathbf{m}_i, \mathbf{s}_k\right)$ is generally nonlinear.

6.2.2 Kalman Filter Localisation

In this course, the procedure for map-based localisation applied at each time step is as follows:

1. Use the robot motion model (state transition model) to estimate the robot pose (1st source of information)
2. Match each measurement with a single landmark in the map. This step is known as data association and it is assumed solved in this course.
3. Use the matched measurement/landmark pair to estimate the robot pose (2nd source of information).
4. Combine the two sources of robot pose using weight average method, e.g., Kalman filter (KF) or Extended Kalman filter (EKF).

Consider a mobile robot that is equipped with an exteroceptive sensor that measures distinguished landmarks in the environment and matched them against different landmarks in the robot internal map that represents the environment.

The robot also knows its motion model. With this setup, the robot has two sources of information about its pose:

1. The robot motion model (state transition model) that guesses where the robot will reside given its previous pose.
2. The observation model that uses exteroceptive sensor to measure the robot pose with respect to the map.

Combining these two poses can be done in a similar manner to that explained in section 6.2.1. Kalman filter and Extended Kalman filter are efficient recursive methods that solve the map-based localisation problem as will be described in this section. Recall the mobile robot with a motion model:

$$\mathbf{s}_k = \mathbf{h}\left(\mathbf{s}_{k-1}, \mathbf{u}_k\right) + \mathbf{q}_k \tag{6.32}$$

where $\mathbf{q}_k \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_k\right)$.

Also, recall the robot observation model:

$$\mathbf{z}_{i,k} = \mathbf{g}\left(\mathbf{m}_i, \mathbf{s}_k\right) + \mathbf{r}_{i,k}, \tag{6.33}$$

where $\mathbf{r}_{i,k} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_k\right)$. If $\mathbf{h}$ and $\mathbf{g}$ are linear functions with respect to their variables, the Kalman filter can be used directly and optimality is guaranteed. Otherwise, Extended Kalman filter is used where both functions $\mathbf{h}$ and $\mathbf{g}$ are linearised around current robot pose estimate using first-order Taylor expansion as follows:

$$\hat{\boldsymbol{\mu}}_k = \mathbf{h}\left(\boldsymbol{\mu}_{k-1}, \mathbf{u}_k\right), \tag{6.34}$$

$$\mathbf{H}_k = \nabla_{\mathbf{s}_{k-1}} \mathbf{h}\left(\mathbf{s}_{k-1}, \mathbf{u}_k\right)\Big|_{\mathbf{s}_{k-1}=\boldsymbol{\mu}_{k-1}}, \tag{6.35}$$

$$\mathbf{s}_k \approx \hat{\boldsymbol{\mu}}_k + \mathbf{H}_k\left(\mathbf{s}_{k-1} - \boldsymbol{\mu}_{k-1}\right). \tag{6.36}$$

$$\hat{\mathbf{z}}_{i,k} = \mathbf{g}\left(\mathbf{m}_i, \hat{\boldsymbol{\mu}}_k\right) \tag{6.37}$$

$$\mathbf{G}_k = \nabla_{\mathbf{s}_k} \mathbf{g}\left(\mathbf{m}_i, \mathbf{s}_k\right)\Big|_{\mathbf{s}_k=\hat{\boldsymbol{\mu}}_k}, \tag{6.38}$$

$$\mathbf{z}_{i,k} \approx \hat{\mathbf{z}}_{i,k} + \mathbf{G}_k\left(\mathbf{s}_k - \hat{\boldsymbol{\mu}}_k\right). \tag{6.39}$$

Based on Extended Kalman filter theory, the aim of the linearised model is to propagate covariance matrices based on Gaussian distributions. Thus, provided that $\mathbf{s}_{k-1} \sim \mathcal{N}\left(\boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1}\right)$ is available, the *prediction step* of the extended Kalman filter is done in a similar manner to motion-based localisation as follows:

$$\hat{\boldsymbol{\mu}}_k = \mathbf{h}\left(\boldsymbol{\mu}_{k-1}, \mathbf{u}_k\right) \tag{6.40}$$

$$\hat{\boldsymbol{\Sigma}}_k = \mathbf{H}_k \boldsymbol{\Sigma}_{k-1} \mathbf{H}_k^T + \mathbf{Q}_k \tag{6.41}$$

Then, the *correction step* of extended Kalman filter is carried out as follows:

$$\hat{\mathbf{z}}_{i,k} = \mathbf{g}(\mathbf{m}_i, \hat{\boldsymbol{\mu}}_k), \tag{6.42}$$

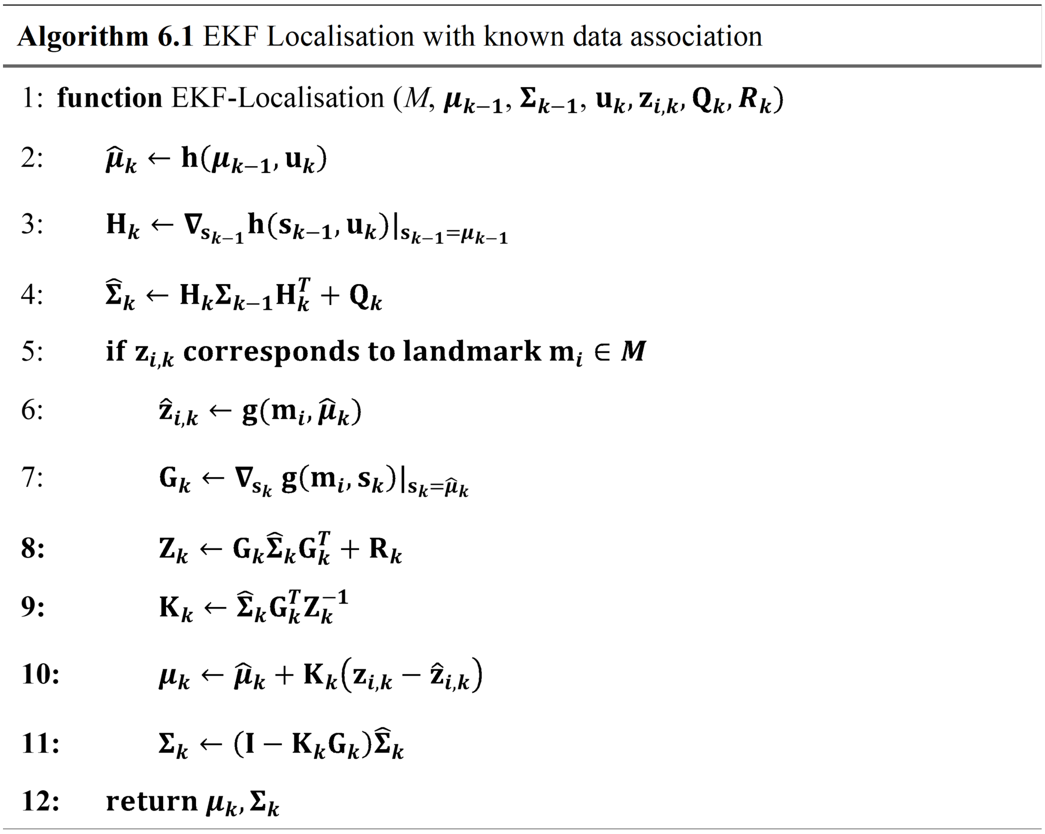
$$\mathbf{Z}_k = \mathbf{G}_k \hat{\boldsymbol{\Sigma}}_k \mathbf{G}_k^T + \mathbf{R}_k, \tag{6.43}$$

$$\mathbf{K}_k = \hat{\boldsymbol{\Sigma}}_k \mathbf{G}_k^T \mathbf{Z}_k^{-1}, \tag{6.44}$$

$$\boldsymbol{\mu}_k = \hat{\boldsymbol{\mu}}_k + \mathbf{K}_k (\mathbf{z}_{i,k} - \hat{\mathbf{z}}_{i,k}), \tag{6.45}$$

$$\boldsymbol{\Sigma}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{G}_k\right) \hat{\boldsymbol{\Sigma}}_k. \tag{6.46}$$

Putting all of the above together, the EKF localisation procedure for known data association is summarised in Algorithm 6.1. The robot initial pose is required for this procedure, otherwise, it will be assumed that $\boldsymbol{\mu}_0 = \mathbf{0}$, and $\boldsymbol{\Sigma}_0 = \mathbf{0}$, i.e., the robot starts at the origin of the world frame $\{W\}$.



Algorithm 6.1 EKF Localisation with known data association

```
function EKF-Localisation ($M$, $\boldsymbol{\mu}_{k-1}$, $\boldsymbol{\Sigma}_{k-1}$, $\mathbf{u}_k$, $\mathbf{z}_{i,k}$, $\mathbf{Q}_k$, $\boldsymbol{R}_k$)
    $\hat{\boldsymbol{\mu}}_k \leftarrow \mathbf{h}(\boldsymbol{\mu}_{k-1}, \mathbf{u}_k)$
    $\mathbf{H}_k \leftarrow \nabla_{\mathbf{s}_{k-1}} \mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)|_{\mathbf{s}_{k-1}=\boldsymbol{\mu}_{k-1}}$
    $\hat{\boldsymbol{\Sigma}}_k \leftarrow \mathbf{H}_k \boldsymbol{\Sigma}_{k-1} \mathbf{H}_k^T + \mathbf{Q}_k$
    if $\mathbf{z}_{i,k}$ corresponds to landmark $\mathbf{m}_i \in \boldsymbol{M}$
        $\hat{\mathbf{z}}_{i,k} \leftarrow \mathbf{g}(\mathbf{m}_i, \hat{\boldsymbol{\mu}}_k)$
        $\mathbf{G}_k \leftarrow \nabla_{\mathbf{s}_k} \mathbf{g}(\mathbf{m}_i, \mathbf{s}_k)|_{\mathbf{s}_k=\hat{\boldsymbol{\mu}}_k}$
        $\mathbf{Z}_k \leftarrow \mathbf{G}_k \hat{\boldsymbol{\Sigma}}_k \mathbf{G}_k^T + \mathbf{R}_k$
        $\mathbf{K}_k \leftarrow \hat{\boldsymbol{\Sigma}}_k \mathbf{G}_k^T \mathbf{Z}_k^{-1}$
        $\boldsymbol{\mu}_k \leftarrow \hat{\boldsymbol{\mu}}_k + \mathbf{K}_k(\mathbf{z}_{i,k} - \hat{\mathbf{z}}_{i,k})$
        $\boldsymbol{\Sigma}_k \leftarrow (\mathbf{I} - \mathbf{K}_k \mathbf{G}_k) \hat{\boldsymbol{\Sigma}}_k$
    return $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$
```

Figure 6.4: EKF Localisation

Kalman filter is a popular method to solve the localisation problem. It is continuous in the state space and its computational complexity is at least quadratic with respect to the state dimension, i.e. $\mathcal{O}\left(n^2\right)$, due to the matrix multiplication in equation (6.44). However, this method requires both the motion and the sensor models to be linear. Also, the noise associated with each model needs to be Gaussian. Kalman filter method looks at the localisation problem as a tracking problem where the initial pose is known with some uncertainty; therefore, Kalman filter does not support multimodal distribution. Due to this fact, Kalman filter localisation cannot solve the *kidnaped robot problem* when the robot collides or moves unexpectedly. Moreover, this method fails when the robot makes a mistake during the data association process, e.g. matching one detected wall with a completely different wall in the map.

6.2.3 Kalman Filter: Case Study

Consider a robot moving in 2D fully observable environment with 6 distinct landmarks as shown in Figure 6.4. The robot pose represents position and orientation of the robot such that $\mathbf{s}_k = \left[s_{x,k} \ s_{y,k} \ s_{\theta,k}\right]^T$, and each landmark is a stationary point feature denoted by $\mathbf{m}_i = \left[m_{x,i} \ m_{y,i}\right]^T$. The robot has two control inputs $\mathbf{u}_k = [v_k \ \omega_k]^T$, where v_k is the robot linear velocity and ω_k is the robot angular velocity. The robot motion model (state transition model) is defined as follows:

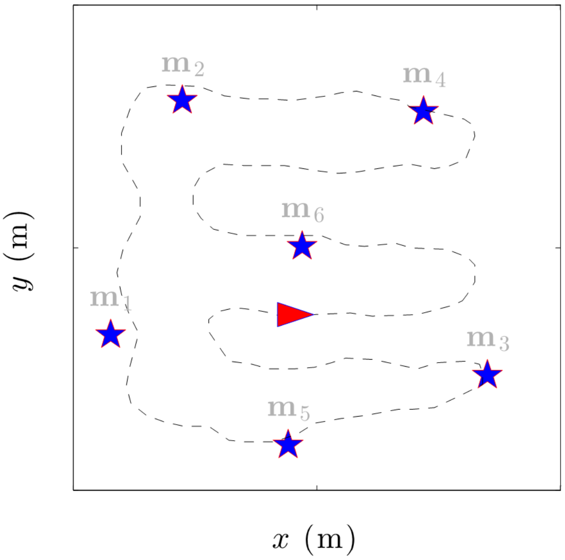


Figure 6.5: Example of mobile robot moving in 2D environment with 6 landmarks represented by the stars, whereas the dashed line represents the robot path. The red triangle represents the initial robot pose, and it points towards the direction of motion.

$$s_{x,k} = s_{x,k-1} + \Delta t.v_k.\cos s_{\theta,k-1} \ + q_{x,k}, \tag{6.47}$$

$$s_{y,k} = s_{y,k-1} + \Delta t.v_k.\sin s_{\theta,k-1} \ + q_{y,k}, \tag{6.48}$$

$$s_{\theta,k} = s_{\theta,k-1} + \Delta t.\omega_k + q_{\theta,k} \tag{6.49}$$

where Δt is the temporal length of consecutive time steps, and the vector $\boldsymbol{q}_k = \left[q_{x,k} \ q_{y,k} \ q_{\theta,k}\right]^T$ represent the motion noise. The robot is equipped with a 360° range-bearing exteroceptive sensor. Reference to principle of operation of the range-bearing sensor illustrated in Figure 6.6, the robot observation model is defined as follows:

$$z_{\rho,i,k} = \sqrt{(m_{x,i} - s_{x,k})^2 + (m_{y,i} - s_{y,k})^2} + r_{\rho,i,k}, \tag{6.50}$$

$$z_{\alpha,i,k} = atan2(m_{y,i} - s_{y,k}, m_{x,i} - s_{x,k}) - s_{\theta,k} + r_{\alpha,i,k} \tag{6.51}$$

where $\mathbf{z}_{i,k} = [\rho_{i,k} \;\; \alpha_{i,k}]^T$ is the measurement vector that consists of landmark angle and bearing in robot frame $\{R\}$, and $\mathbf{r}_{i,k} = [r_{\rho,i,k} \;\; r_{\alpha,i,k}]^T$ is the measurement noise vector. The noise of both motion and observation models is Gaussian.

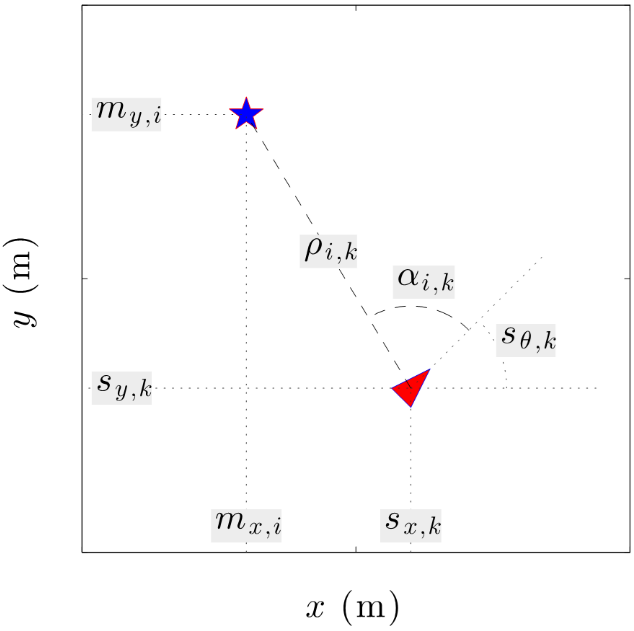


Figure 6.6: Observation model for a mobile robot in 2-D environment at time step k. The triangle is the robot pose $\mathbf{s}_k$, and the star is the landmark $\mathbf{m}_i$, and both are with respect to the world reference frame $\{W\}$, whereas the measurements $\rho_{i,k}$, $\alpha_{i,k}$ are in the robot frame $\{R\}$.

Since the robot motion and observation models are nonlinear, EKF is used where the Jacobian matrix $\mathbf{H}_k$ is defined in equation (6.10). The Jacobian of the observation model (the matrix $\mathbf{G}_k$) is computed as follows:

$$\Delta x = m_{x,i} - \hat{s}_{x,k}, \tag{6.52}$$

$$\Delta y = m_{y,i} - \hat{s}_{y,k}, \tag{6.53}$$

$$p = \Delta x^2 + \Delta y^2, \tag{6.54}$$

$$\mathbf{G}_k = \begin{bmatrix} \frac{\partial g_1}{\partial s_{x,k}} & \frac{\partial g_1}{\partial s_{y,k}} & \frac{\partial g_1}{\partial s_{\theta,k}} \\ \frac{\partial g_2}{\partial s_{x,k}} & \frac{\partial g_2}{\partial s_{y,k}} & \frac{\partial g_2}{\partial s_{\theta,k}} \end{bmatrix} = \begin{bmatrix} -\frac{\Delta x}{\sqrt{p}} & -\frac{\Delta y}{\sqrt{p}} & 0 \\ \frac{\Delta y}{p} & -\frac{\Delta x}{p} & -1 \end{bmatrix}. \tag{6.55}$$

For this example, the prediction step and correction step of EKF follow the same equations described in section (6.2.2). Figure 6.5 shows the result of EKF localisation applied to the example illustrated in Figure 6.7.

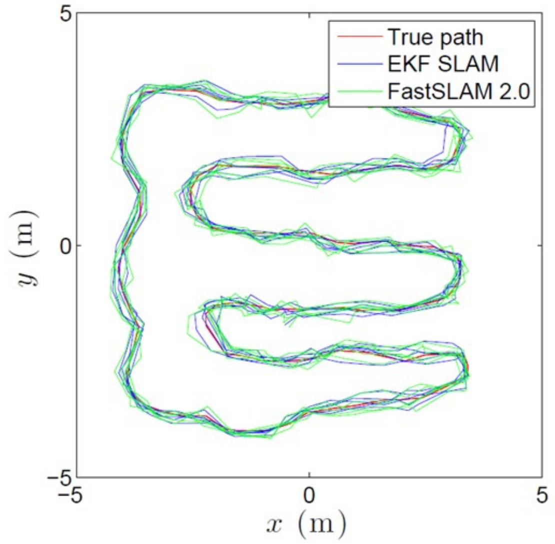


Figure 6.7: Estimated robot pose after applying EKF localisation.

6.2.4 Kalman Filter: Practical Considerations

It is important to handle angles carefully and consider the wrapping effect. For example, consider the angle x in radians; from the point of view of the robot orientation, this angle is the same as $x + 2n\pi$. Equations (6.25), (6.32), and (6.38) show that the resultant angle can exceed the $[-\pi : \pi]$ limit. This can lead to wrong estimation of the mean $\hat{\boldsymbol{\mu}}_k$ in the correction step in EKF. To avoid this problem, it is important to wrap these angles and limit them to values between $-\pi$ and π. In Matlab, this is done using "*wrapToPi*" function.

6.2.5 Kalman Filter: Numerical Example

Map based localisation

A non-holonomic robot navigates in a partial unknown environment.

GIVEN	FIND
$\boldsymbol{\mu}_0 = \begin{bmatrix} s_{x,0} \\ s_{y,0} \\ s_{\theta,0} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ robot initial position $\boldsymbol{\Sigma}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ initial covariance matrix $\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ landmark position Assume the following conditions remain constant for all $k > 0$ $\boldsymbol{Q_k} = \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix}$ motion model covariance matrix $\boldsymbol{R_k} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.02 \end{bmatrix}$ observation model covariance matrix $V_k = 1 meter/second$, mobile robot linear velocity $\omega_k = 1 radian/second$, mobile robot angular velocity $\Delta t = 0.1 seconds$, sampling time Assumed measurements at each step k $\boldsymbol{z}_{1,1} = [4.87 \ \ 0.8]'$ $\boldsymbol{z}_{1,2} = [4.72 \ \ 0.72]'$ $\boldsymbol{z}_{1,3} = [4.69 \ \ 0.65]'$	Using Kalman filter estimate the position of the robot for three time steps, i.e., μ_1 , μ_2 , μ_3 and Σ_1 , Σ_2 , Σ_3 .