1

KALMAN FILTER MAPPING

Please Notice: there is a technical error in this chapter: the liberalization is usually not performed with respect to the time t but with respect to the state variables. This will be corrected in a forthcoming revision

1.1 MOTIVATION

The previous chapter described algorithms for mapping with *known* poses. Let us now return to the concurrent mapping and localization problem, in which the poses are not given. Recall that a key difficulty in mapping is that the robot accrues odometry error while gathering data. This produces a problem of concurrently estimating a map and the robot's poses. This combined problem is much harder than the mapping problem with known poses studied in the previous chapter, since it involves a hole new source of uncertainty that can grow without bounds.

In this chapter, we describe a family of algorithms that are commonly known as Kalman filter mapping algorithms, and often referred to as *SLAM* algorithms. SLAM stands for *simultaneous localization and mapping*, which is a different name for concurrent mapping and localization. While technically, SLAM characterizes a problem (the mapping problem) and not a specific solution, the term SLAM is often identified with a specific style of approach that will be discussed in this chapter.

The algorithms described in this chapter are based on the same basic idea as the localization algorithms described in Chapter **??**. The key idea is to estimate a

posterior probability. In the case of localization, the posterior was defined over the robot's pose, which is a three-dimensional quantity. In concurrent mapping and localization, the state space is much higher dimensional. In addition to the robot's pose, it includes the environment itself. Nevertheless, under certain restrictive assumptions, it is in fact possible to estimate the posteriors jointly over poses and maps.

Estimating the full posterior over poses and maps is, probabilistically speaking, the gold standard. The posterior not only contains information about the most probable map and pose, but also about the residual uncertainty in the estimate, which can be informative when using the map for robot navigation. Unfortunately, estimating the full posterior over maps and poses is difficult, due to the high dimensionality of the space of all maps. In the localization problems studied earlier in this book, we developed algorithms for full posterior estimation over robot poses. Poses are defined over a three-dimensional space, whereas the typical maps studied in robotics have between hundreds and millions of parameters. Therefore, many of the basic estimators are inapplicable.

The family of algorithms studied in this chapter use Kalman filters for posterior estimation. Kalman filters were already discussed in Chapter **??**, as a special version of Bayes filters. As discussed there, Kalman filters represent posteriors by multivariate Gaussians. To maintain a Gaussian belief, Kalman filters make a range of restrictive assumptions which are inherited by the algorithms described in this chapter. With regards to the concurrent mapping and localization problem, these restrictions are as follows:

- **Landmarks.** The Kalman filter approach to robot mapping requires that maps are collections of identifiable point landmarks. Point landmarks are landmarks that are confined to a small physical location in the world. Mapping the environment is equivalent to determining the number of landmarks and determining their coordinates relative to a global coordinate frame. One of the reasons for representing the map by a collection of landmarks lies in the computational complexity. The basic Kalman filter approach scales quadratically with the size of the map. Feature sets of more than 100 features are therefore difficult to handle in real-time. Another and more important reason for using landmarks stems from the Gaussian noise assumption, which will be discussed further below.

- **Linearity.** The algorithms discussed here assume linear models of motion and perception. This is a direct consequence of the basic Kalman filter algorithm. The motion and perceptual models discussed in Chapter **??**, in contrast, are all non-linear. However, as long as the robot's uncertainty

is small, this is not too severe an assumption, as these non-linear models can be approximated by linear ones. The resulting filter, often referred to as *extended Kalman filter* (EKF), is well-understood and converges to the correct filter for the non-linear estimation problem as the subsampling rate increases, under very mild conditions.

- **Gaussian noise.** The non-determinism in robot motion and perception is assumed to be the effect of independent, Gaussian noise. This restriction might look acceptable, but in fact it severely limits the type environment features that can be mapped. Why? Consider two identical landmarks. If the robot observes one of them, the resulting pose estimate will have two modes, one for each of the two identical landmarks. Such cases cannot be handled by the Kalman filter approach. The problem is often referred to as *data association problem*, or *correspondence problem*. The data association problem is the problem of associating landmark sightings to actual landmarks in the world. If landmarks can be identified uniquely, there is no data association problem. Sometimes, landmarks are spaced far enough apart that their identity can be determined with sufficient accuract. In other environments, this may not be the case and the robot faces a data association problem. Algorithms that can cope with unknown data associations will be discussed in the two chapters following the present one. For now, we will simply assume that the problem does not exist. However, we note that the Gaussian noise assumption and the implied lack of a data association problem is by far the most severe practical limitation of present Kalman filtering algorithms. We also note that some recent research has shown promise in overcoming this assumption.

- **Positive measurements only.** Another consequence of the Gaussian noise assumption is an inability to incorporate the *absence* of a landmark into a robot's belief. In general, the robot can acquire information about its environment by two types of events: Detecting a landmark and not detecting a landmark. Clearly, not seeing a landmark carries information about the environment. However, the the posterior corresponding to such measurements is non-Gaussian, and in fact the underlying measurement model would inevitably violate the Gaussian noise assumption. The algorithms discussed here are therefore unable to incorporate such information, and simply discard it.

As this list suggests, the algorithms described in this chapter make severe restriction on the nature of the model and the available sensor data. The by far most severe assumption is the lack of a *data association problem*. On the positive side, when these algorithms are applicable, they are usually superior to any

alternative technique. To see why, we should remind ourselves that estimating the full posterior over the quantities of interest is the most one can hope for. Any of the algorithms discussed in the subsequent chapters are much weaker, in that they find only a single map that cannot be guaranteed to be optimal.

This chapter is organized as follows. First, we will extend the Kalman filter to the simultaneous localization and mapping problem, by introducing an augmented state representation that comprises maps and robot poses. Much of this material builds on Chapter **??**, where the basic Kalman filter was introduced. Subsequently, we will show how to linearize the motion model and the perceptual model for commonly used landmark detectors. This will lead to the first algorithm, which implements Kalman filtering for a known numbers of landmarks. We will describe a maximum likelihood strategy for coping with the data association problem that works well when similar looking landmarks are sufficiently far apart. Finally, we will provide techniques for handling important practical issues, such as outlier removal and variable number of landmarks. Pointers to extensions that build on this framework will be provided towards the end of this chapter.

1.2 KALMAN FILTER ESTIMATION

1.2.1 The Augmented State Model

Let us begin with the very basics of Kalman filter mapping. The first definition regards the nature of the state that is being estimated. In the concurrent mapping and localization problem, there exist two types of unknown variables that systematically affect sensor readings: The robot's poses, and the environment itself, which is also unknown. Thus, the state vector comprises the robot's pose s and the map m. From now on, let us denote this state by a, and write:

$$a = \begin{pmatrix} s \\ m \end{pmatrix} \tag{1.1}$$

The state a is often referred to as the *augmented* state, to distinguish it from the robot pose, which is considered the state in mobile robot localization. Clearly, the augmented state can have many thousand dimensions. In fact, at the beginning of mapping, the size of the augmented state vector is usually unknown to the robot. Instead, the robot has to dynamically estimate its size during mapping. This is a key difference to the localization problem, where the dimensionality of the state was fixed throughout the problem. However, for now

we will assume that we are given the number of landmarks in the environment and hence know the dimensionality of the augmented state. We will revisit the issue of estimating the size of the augmented state vector after introducing the basic Kalman filter mechanisms.

Let us now make the elements of the augmented state a a bit more explicit. As before, we describe the robot's pose s by three variables

$$s = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{1.2}$$

where x and y are the coordinates of the robot in the plane, and θ is the robot's heading direction. As usual, the pose at time t will be denoted s_t, and the set of all poses in the interval $[t_0, t_1]$ will be denoted $s_{t_0 \ldots t_1}$.

In Kalman filter mapping, it is useful to think about the map m as a collection of features, or landmarks. Each landmark possesses a location relative to the global coordinate system. In our exposition, landmarks do not possess an orientation; hence, two variables are needed to specify a landmark location. For the i-th landmark, let us denote these coordinates using the variables x_i and y_i. The actual number of landmarks in the environment will be denoted by N. For now, let us assume we are told the number of landmarks in the environment. The map m is then the following $2N$-dimensional vector:

$$m = \begin{pmatrix} x_1 & y_1 & x_2 & y_2 & \ldots & x_N & y_N \end{pmatrix}^T \tag{1.3}$$

Here x_i and y_i are the coordinates of the i-the landmark. We use the transpose ^{T} to save space.

The augmented state vector a comprises the robot pose s and the map m. The robot pose was specified in Equation (1.2), and the map in Equation and (1.3). Putting both together gives us:

$$a = \begin{pmatrix} x & y & \theta & x_1 & y_1 & \ldots & x_N & y_N \end{pmatrix}^T \tag{1.4}$$

Obviously, the augmented state is a vector of dimension $2N+3$, where N is the number of landmarks in the map. For now, we will assume that N is known. The more plausible case where the number of landmarks N is unknown will be discussed further below in this chapter.

1.2.2 Bayes Filters in Mapping

The definition of the augmented state vector makes it possible—at least in principle—to apply Bayes filters to this estimation problem. To do so, we have to convince ourselves that the augmented state a is the complete state of the environment, in the sense that it fulfill the Markov assumption that underlies the derivation of the Bayes filters. A common assumption in robotic mapping is that the environment is static, with the exception of the robot, which may move about the environment. In such environments, knowledge of the (exact) map the robot's pose indeed renders the past and the future independent. Put differently, if one is told the exact map and robot pose, past sensor measurements do not contain any additional information about future measurements. Thus, the augmented state is indeed sufficient, and Bayes filters are applicable.

The basic Bayes filter for posterior estimation was already discussed extensively in Chapter **??**, and is here restated for the reader's convenience using the augmented state a:

$$b_t(a_t) \quad = \quad \eta \, p(z_t \mid a_t) \int \int p(a_t \mid u_{t-1}, a_{t-1}) \, b_t(a_{t-1}) \, da_{t-1} \tag{1.5}$$

Finding a technique for estimating the belief b_t is the central objective of this chapter.

Unfortunately, many of the basic filter algorithms in Chapter **??** are not directly applicable to the problem of estimating posteriors over a, simply because a possesses too many dimensions. For example, particle filters require an exponential number of particles to densely populate a space, which is infeasible for any practical number of landmarks N.

Luckily, one of the variants of Bayes filters discussed in Chapter **??** is applicable despite the high dimensionality of the augmented state a: Kalman filters. As discussed in Chapter **??**, Kalman filters represent the posterior by a multivariate Gaussian distribution. We will denote the mean of this Gaussian by μ and the covariance matrix by Σ. The posterior distribution is then given by the following mean vector μ and covariance matrix Σ. Here, the indexes x, y, and θ correspond to the robot's pose $s = \langle x, y, \theta \rangle$, and indexes with a number (x_i, y_i) correspond to the coordinates of the i-th feature or landmark.

$$\mu \quad = \quad \begin{pmatrix} \mu_x & \mu_y & \mu_\theta & \mu_{x_1} & \mu_{y_1} & \mu_{x_2} & \mu_{y_2} & \cdots & \mu_{x_N} & \mu_{y_N} \end{pmatrix}^T \tag{1.6}$$

$$\Sigma = \begin{pmatrix} \sigma_{x,x} & \sigma_{y,x} & \sigma_{\theta,x} & \sigma_{x_1,x} & \sigma_{y_1,x} & \cdots & \sigma_{x_N,x} & \sigma_{y_N,x} \\ \sigma_{x,y} & \sigma_{y,y} & \sigma_{\theta,y} & \sigma_{x_1,y} & \sigma_{y_1,y} & \cdots & \sigma_{x_N,y} & \sigma_{y_N,y} \\ \sigma_{x,\theta} & \sigma_{y,\theta} & \sigma_{\theta,\theta} & \sigma_{x_1,\theta} & \sigma_{y_1,\theta} & \cdots & \sigma_{x_N,\theta} & \sigma_{y_N,\theta} \\ \sigma_{x,x_1} & \sigma_{y,x_1} & \sigma_{\theta,x_1} & \sigma_{x_1,x_1} & \sigma_{y_1,x_1} & \cdots & \sigma_{x_N,x_1} & \sigma_{y_N,x_1} \\ \sigma_{x,y_1} & \sigma_{y,y_1} & \sigma_{\theta,y_1} & \sigma_{x_1,y_1} & \sigma_{y_1,y_1} & \cdots & \sigma_{x_N,y_1} & \sigma_{y_N,y_1} \\ \sigma_{x,x_2} & \sigma_{y,x_2} & \sigma_{\theta,x_2} & \sigma_{x_1,x_2} & \sigma_{y_1,x_2} & \cdots & \sigma_{x_N,x_2} & \sigma_{y_N,x_2} \\ \sigma_{x,y_2} & \sigma_{y,y_2} & \sigma_{\theta,y_2} & \sigma_{x_1,y_2} & \sigma_{y_1,y_2} & \cdots & \sigma_{x_N,y_2} & \sigma_{y_N,y_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{x,x_N} & \sigma_{y,x_N} & \sigma_{\theta,x_N} & \sigma_{x_1,x_N} & \sigma_{y_1,x_N} & \cdots & \sigma_{x_N,x_N} & \sigma_{y_N,x_N} \\ \sigma_{x,y_N} & \sigma_{y,y_N} & \sigma_{\theta,y_N} & \sigma_{x_1,y_N} & \sigma_{y_1,y_N} & \cdots & \sigma_{x_N,y_N} & \sigma_{y_N,y_N} \end{pmatrix}$$

The mean vector μ contains $2N+3$ values, just as the augmented state vector a. The much larger covariance matrix possesses $(2N+3)^2 = 4N^2+12N+9$ values. However, covariance matrices are symmetric. It therefore suffices to memorize $\frac{1}{2}[(2N+3)^2+2N+3] = 2N^2+7N+6$ values. For $N=100$ landmarks, the mean is represented by 203 values and the covariance by 20,706 values. These numbers might appear relatively small. However, updating such large matrices involves the multiplication of covariance matrices, which is an expensive operation.

It will be convenient to annotate the robot's belief with the time index t. If the robot's belief at time t is given by μ_t and Σ_t, the robot's best estimate of its own position is given by $\langle \mu_{x,t}, \mu_{y,t}, \mu_{\theta,t} \rangle$. The robot's best estimate of the coordinates of the i-th landmark is $\langle \mu_{x_i,t} \mu_{y_i,t} \rangle$. The uncertainty in these estimate is expressed by the covariance matrix Σ_t. For example, $\sigma_{x,x,t}$ is the variance of the robot's estimate of its own x-coordinate at time t, and $\sigma_{x_i,x_i,t}$ is the variance of the estimate of the x-coordinate of the i-th landmark at time t. Of great importance are the off-diagonal terms in the covariance matrix, which measure the covariance between different variables. The off-diagonal elements can be subdivided into three groups: Those that only apply to robot coordinates (such as: $\sigma_{x,\theta},t$), those that only apply to landmark coordinates (such as: $\sigma_{x_i,y_j,t}$ for $1 \leq i,j \leq N$), and those that measure the covariance between robot and landmark coordinates. In general, all three types of covariances are non-zero. This is because the errors in the robot pose estimates and the landmark coordinate estimates are all correlated. To see, imagine a robot that suffers a significant error in its odometry. This error not only affects its own pose estimate, but also the estimates of the coordinates of all landmarks that the robot observes afterwards. Since this affect is systematic in that it may be caused by a single odometric error, all those estimation errors are correlated. In Kalman filtering, the covariances in Σ_t are therefore important. One may, for example, not just use the main diagonal of the covariance matrices (the variances) for mapping, in hope of simplifying the computation of the map.

Instead, all elements of the covariance matrix matter, and they all have to be calculated when building maps.

Armed with a representation of the posterior, we can now formulate the basic algorithm that implements Bayes filters. As stated in Chapter **??**, Kalman filters possess a convenient update equation that involves the vector μ and the covariance matrix Σ. For the reader's convenience, let us restate basic update equation:

if d_t *is a perceptual data item* z_t *then*
$K_t = \Sigma_t C_t^T (C_t \Sigma_t C_t^T + \Sigma_{z,t})^{-1}$
$\mu_{t+1} = \mu_t + K_t(z_t - C_t \mu_t)$
$\Sigma_{t+1} = (I - K_t C_t)\Sigma_t$

else if d_t *is an action data item* u_t *then*
$\mu_{t+1} = A_t \mu_t + B_t u_t$
$\Sigma_{t+1} = A_t \Sigma_t A_t^T + \Sigma_{u,t}$

return $\langle \mu_{t+1}, \Sigma_{t+1} \rangle$

Obviously, we are lacking critical information to implement this filter, In particular, we do not yet know how to design the matrices A_t, B_t, C_t, $\Sigma_{z,t}$ and $\Sigma_{u,t}$. The matrices C_t and $\Sigma_{z,t}$ are synonym for the robot's measurement model, $p(z_t|y_t, m_t)$, defined over the augmented state y_t. A_t, B_t, and $\Sigma_{u,t}$ describe the robot's motion model, $p(y_{t+1}|u_t, y_t)$, also defined over the augmented state y_t. The next two sections will explain how to obtain these matrices.

We also notice that there are two expensive operations in the Kalman filer: A matrix inversion, and a matrix multiplication. In state-of-the-art implementations, the costs of matrix multiplication dominates that of the inversion, since the matrices being multiplied are much larger than those that have to be inverted. However, let us now turn our attention to calculating the missing matrices A_t, B_t, C_t, $\Sigma_{z,t}$ and $\Sigma_{u,t}$.

1.2.3 Linearized Motion Model

In the general Bayes filter, the motion model is given by the conditional probility $p(s_{t+1}|u_t, s_t)$. In Chapter **??**, we already discussed in depth two such

probabilistic motion models, one using odometry coordinates and one based on robot velocities. Unfortunately, neither of these models are fit for the Kalman filtering algorithm for concurrent mapping and localization. This is because of two reasons: First, any motion model in the Kalman filter approach has to be defined over the augmented state a_t, instead the robot pose s_t. Second, Kalman filters require that motion models are linear with additive Gaussian noise—wheras the motion models in Chapter **??** were all non-linear.

Let us look at these two requirements in detail. Generalizing a motion model to the augmented state a_t requires us to define a conditional probabilty density $p(a_{t+1}|u_t, a_t)$, where u_t is a control and a_{t+1} and a_t are augmented states at time $t+1$ and time t, respectively. We note that one of the key assumptions in robot mapping is that the environment does not change over time. Hence, controls have no effect on the map m_t. Consequently, the posterior probability $p(m_{t+1}|u_t, m_t)$ of the map m_{t+1} after executing action u_t is a point-mass distribution centered on the map $m_t = m_{t+1}$:

$$p(m_{t+1}|u_t, m_t) \quad = \quad I_{m_{t+1}=m_t} \tag{1.7}$$

Here I is the indicator function, which assumes the value 1 if and only if its logical argument is true. Putting this together with the regular robot moton model $p(s_{t+1}|u_t, s_t)$ gives us the desired motion model over the augmented state space:

$$\begin{aligned} p(a_{t+1}|u_t, a_t) &= p(m_{t+1}, s_{t+1}|u_t, m_t, s_t) \\ &= I_{m_{t+1}=m_t}\ p(s_{t+1}|u_t, s_t) \end{aligned} \tag{1.8}$$

The second restriction, hoewever, is more severe: Kalman filters require that motion models are *linear with additive Gaussian noise.* This requirement is fulfilled if we can write the robot motion via the following stochastic equation:

$$a_{t+1} \quad = \quad A_t a_t + B_t u_t + \varepsilon_{u,t} \tag{1.9}$$

Here a_t is the present state, u_t is the control, and a_{t+1} is the succeeding state after executing the control. A_t and B_t are matrices. A_t is a quadratic matrix of size $(2N+3)^2$, which maps augmented states a_t to successor states a_{t+1}. B_t is a matrix of dimension $2N+3$ times the dimension of the control variable u_t, which throughout this chapter will be of dimension two (see Chapter **??**). Finally, the variable ε_u is a $(2N+3)$-dimensional noise variable, which is a Gaussian random variable with zero mean and covariance $\Sigma_{u,t}$.

We can restate Equation (1.9) as a probability density, exploiting the Gaussian nature of the noise variable $\varepsilon_{u,t}$:

$$p(a_{t+1}|u_t, a_t) = \frac{1}{\sqrt{(2\pi)^{2N+3} \mid \Sigma_{u,t} \mid}} e^{-\frac{1}{2}(A_t a_t + B_t u_t - a_{t+1})^T \Sigma_{u,t}^{-1} (A_t a_t + B_t u_t - a_{t+1})} \tag{1.10}$$

Written in this form, it should be obvious that the motion model is a linear Gaussian model with mean $A_t a_t + B_t u_t$ and covariance $\Sigma_{u,t}$.

Before addressing the problem of designing linear-Gaussian motion models in its full glory, we notice that one of those matrices is trivially estimated: The matrix A_t. To see, consider the degenerate where the robot does not move. Here the controls u_t are zero, and hence is the term $B_t u_t$ in Equation (1.9). In the absence of robot motion, none of the environment state variables changes, i.e., $a_t = a_{t+1}$. Hence, A_t must be the *identity matrix*:

$$A_t = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{1.11}$$

This leaves us with the problem of detrmining the matrices B_t and $\Sigma_{u,t}$.

To calculate those matrices, we notice they also possess a special form. In partixularly, both of these matrices contain only a small number of non-zero elements. This is because in static environents, the location of landmarks is neither influenced by robot motion, nor by the noise variable $\Sigma_{u,t}$.

To make this precise, let us decompose the matrices B_t and $\Sigma_{u,t}$ into two components each. One of these components corresponds to the robot pose s, and will be referred to as $B_{s,t}$ and $\Sigma_{u,s,t}$, respectively. The other component corresponds to the map m, and will be denoted $B_{m,t}$ and $\Sigma_{u,m,t}$, respectively:

$$B_t = \begin{pmatrix} B_{s,t} \\ B_{m,t} \end{pmatrix} \tag{1.12}$$

$$\Sigma_{u,t} = \begin{pmatrix} & & & 0 & \cdots & 0 \\ & \Sigma_{u,s,t} & & \vdots & \ddots & \vdots \\ & & & 0 & \cdots & 0 \\ 0 & \cdots & 0 & & & \\ \vdots & \ddots & \vdots & & \Sigma_{u,m,t} & \\ 0 & \cdots & 0 & & & \end{pmatrix} \tag{1.13}$$

Under the assumption that the environment does not change during mapping, the matrices $B_{m,t}$ and $\Sigma_{u,m,t}$ are null matrices. These considerations cover all trivial aspects of designing the motion matrices A_t, B_t, and $\Sigma_{u,t}$. What remains to be defined are two much smaller sub-matrices: The matrix $B_{s,t}$, which is a 3 by 2 marix that specifies the effect of robot controls u_t on the robot pose s_t, and the matrix $\Sigma_{u,m,t}$, a matrix of size 3 by 3 which defines the noise in robot motion.

Unfortunately, those two matrices are more difficult to define. The difficulty arises from the fact that robot motion is non-linear. For example, a robot that moves with constant translational and rotational velocity typically moves on a circular trajectory, which cannot be described by a linear equation. This is at odds with the fact Kalman filtering requires linear motion models, as manifested by the matrices discussed in this section. At first glance, this seems to render Kalman filtering inapplicable for state estimation in mobile robotics.

However, there is an approximation that works well in practice. The common approach to Kalman filtering in non-linear domain is to *linearize* the motion model. The basic idea behind linearizing a function $f(x)$ is that for any point x_0, any differentiable function f can be approximated by a linear function defined through the value of the function $f(x_0)$ and its first derivative $f'(x_0)$ at x_0. Since the robot motion model is differentiable, we can (and will) linearize it.

To do so, let us adopt the velocity-based motion model defined in Chapter **??** (page **??**) of this book. The reader may recall that the velocity-based motion model assumes that the robot is controlled via two independent velocity, a translational velocity denoited v_t, and a rotational velocity denoted ω_t. A mathematical analysis of the motion with fixed velocities led to the algorithm **sample_motion_model_velocity** summarized in Table **??** on page **??**.

Let us briefly restate the main motion equations drived in Chapter **??**, to the extent necessary for deriving the linear approximation suitable for Kalman filter mapping. As stated above, the control is a vector of two independent velocities:

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} \tag{1.14}$$

where v_t is a translaitonal velocity, and ω_t is a rotational velocity (and T denotes the transpose). The *noise-free* motion is governed by the following equations, which is copied directly from Equations (**??**) through (**??**). Here Δt is a time

window over which the robot motion is predicted:

$$x_{t+1} = x_t - \frac{v_t}{\omega_t}\sin\theta_t + \frac{v_t}{\omega_t}\sin(\theta_t + \omega_t \Delta t) \tag{1.15}$$

$$y_{t+1} = y_t + \frac{v_t}{\omega_t}\cos\theta_t - \frac{v_t}{\omega_t}\cos(\theta_t + \omega_t \Delta t) \tag{1.16}$$

$$\theta_{t+1} = \theta_t + \omega_t \Delta t \tag{1.17}$$

The linear approximation is given by the following approximation:

$$s_{t+1} = s_t + \Delta t\ \dot{s}_t \tag{1.18}$$

or, broken down into pose coordinate variables,

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t + \Delta t\ \dot{x}_t \\ y_t + \Delta t\ \dot{y}_t \\ \theta_t + \Delta t\ \dot{\theta}_t \end{pmatrix} \tag{1.19}$$

Here $\dot{x}_t$, $\dot{y}_t$, and $\dot{\theta}_t$ denote the first derivative of the pose variables x, y, and θ with respect to the time variable, and calculated at the robot's pose at time t. Put differently, the linearized motion model is one that only models first order effects, by linearizing the non-linear motion model at the current robot pose s_t. Such an operation is known as a Taylor-series expansion of degree 1.

In the context of our velocity-based motion model, we recall that in the noise-free case the robot's trajectory describes a circle with radius $\frac{v_t}{\omega_t}$. Our linear approximation suggests that the robot instead moves along a straight line, which is tangent to the circle at the pose s_t at time t. One might think that a line is a poor approximation to a circle, and in fact it is, but only for large values of Δt. For very small time intervals Δt, the approximation can be sufficiently accurate. Thus, our linear approaximation is valid if the time interval Δt is sufficiently small. The larger the time interval Δt, the more the linear approximation and the non-linear motion model diverge, hence the less accurate the result.

Let us now make the linearized motion model concrete for our velocity-based motion model, by calculating the derivatives $\dot{x}$, $\dot{y}$, and $\dot{\theta}$. These derivatives are directly obtained from Equations (1.15) through (1.17). In particular, the first derivative is as follows:

$$\begin{aligned} \dot{x}_t &= \left.\frac{\partial x_{t+1}}{\partial \Delta t}\right|_{\Delta t=0} \\ &= \left.\frac{\partial}{\partial \Delta t}\left[x_t - \frac{v_t}{\omega_t}\sin\theta_t + \frac{v_t}{\omega_t}\sin(\theta_t + \omega_t \Delta t)\right]\right|_{\Delta t=0} \end{aligned}$$

$$
\begin{aligned}
&= \frac{v_t}{\omega_t} \cos(\theta_t + \omega_t \Delta t)\ \omega_t \Big|_{\Delta t=0} \\
&= v_t \cos(\theta_t + \omega_t \Delta t)|_{\Delta t=0} \\
&= v_t \cos\theta_t
\end{aligned} \tag{1.20}
$$

A similar derivation gives us for the two remaining derivatives:

$$\dot{y}_t = v_t \sin\theta_t \tag{1.21}$$

$$\dot{\theta}_t = \omega_t \tag{1.22}$$

To summarize, we obtain the derivatives

$$\dot{s}_t = \begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{\theta}_t \end{pmatrix} = \begin{pmatrix} v_t \cos\theta_t \\ v_t \sin\theta_t \\ \omega_t \end{pmatrix} \tag{1.23}$$

Substituting these derivatives back into Equation (1.19) gives us the linear approximation to the velocity-based motion model defined in Chapter **??**, and used throughout this book:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t + \Delta t\ v_t \cos\theta_t \\ y_t + \Delta t\ v_t \sin\theta_t \\ \theta_t + \Delta t\ \omega_t \end{pmatrix} \tag{1.24}$$

As noticed above, our linear approximation is reasonable if Δt is small, exploiting the continuous nature of robot motion. In fact, Kalman filters that employ linerarizations such as the one derived here are commonly known as *extended Kalman filters (EKFs)*. In EKFs, one often subdivides the time unit into multiple sub-intervals. The finer the subdivision, the smaller the error introduced by the linearization. The linearization also explains why we annotated the various matrices by the time index t. Clearly, the matrix $B_{s,t}$ is a function of the momentary robot heading θ_t, hence it changes over time. So while the individual motion models are linear, the composition of them is clearly non-linear, approximating the non-linear nature of robot motion.

The biggest omission in our consideration so far is the inherent *noise* in robot motion. The nature of the noise was discussed in length in Chapter **??**; clearly, its effect on the robot pose variable is non-Gaussian, even if the velocity noise is. The linearized motion model simply approximates this complex noise by an additive Gaussian term. That is, the noisy motion is approximated by the following expression, where ε_{x_t}, ε_{y_t}, and ε_{θ_t} are independent Gaussian noise variables with zero mean:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t + \Delta t\ v_t \cos\theta_t & +\varepsilon_{x_t} \\ y_t + \Delta t\ v_t \sin\theta_t & +\varepsilon_{y_t} \\ \theta_t + \Delta t\ \omega_t & +\varepsilon_{\theta_t} \end{pmatrix} \tag{1.25}$$

We will denote the variances of the three noise variables by σ_{x_t}, σ_{y_t}, and σ_{θ_t}, respectively.

We are now in the position to define the linear matrices $B_{s,t}$ and $\Sigma_{u,s,t}$, which completes our derivation of the linear-Gaussian motion model in the Kalman filter approach. In particular, we have

$$B_{s,t} = \begin{pmatrix} \Delta t \cos\theta_t & 0 \\ \Delta t \sin\theta_t & 0 \\ 0 & \Delta t \end{pmatrix} \tag{1.26}$$

and

$$\Sigma_{u,s,t} = \begin{pmatrix} \sigma_{x_t} & 0 & 0 \\ 0 & \sigma_{y_t} & 0 \\ 0 & 0 & \sigma_{\theta_t} \end{pmatrix} \tag{1.27}$$

If we now for convenience write

$$\varepsilon_{s_t} = \begin{pmatrix} \varepsilon_{x_t} \\ \varepsilon_{y_t} \\ \varepsilon_{\theta_t} \end{pmatrix} \tag{1.28}$$

we can write the linear-Gaussian robot motion model (1.25) in the conveninent matrix form:

$$s_{t+1} = I + B_{s,t} u_t + \varepsilon_{s_t} \tag{1.29}$$

is indeed equivalent to the linearized motion model described in (1.25). This equation, obviously, only characterizes the effect of controls on the robot pose s_t, which is only one component of the augmented state y_t.

Putting everything together, we obtain the desired action model for the augmented state a_t:

$$\underbrace{\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \\ x_{1,t+1}, \\ y_{1,t+1} \\ \vdots \\ x_{N,t+1} \\ y_{N,t+1} \end{pmatrix}}_{a_{t+1}} = \underbrace{\begin{pmatrix} x_t \\ y_t \\ \theta_t \\ x_{1,t}, \\ y_{1,t} \\ \vdots \\ x_{N,t} \\ y_{N,t} \end{pmatrix}}_{a_t} + \underbrace{\begin{pmatrix} \Delta t \cos\theta_t & 0 \\ \Delta t \sin\theta_t & 0 \\ 0 & \Delta t \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 0 \end{pmatrix}}_{B_{u,t}} \cdot \underbrace{\begin{pmatrix} v_t \\ \omega_t \end{pmatrix}}_{u_t} + \underbrace{\begin{pmatrix} \varepsilon_{x_t} \\ \varepsilon_{y_t} \\ \varepsilon_{\theta_t} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}}_{\varepsilon_{u,t}} \tag{1.30}$$

Notice that we silently omitted the identity matrix A_t, exploiting the fact that A_t is the identity matrix. We also silently ignored that the amount of noise in motion Σ_u depends on the magnitude of robot motion u_t. When implementing Kalman filter mapping, the covariance matrix Σ_u is usually a function of the velocities v_t and ω_t. This issue was already discussed in more depth in Chapter **??**, and for the sake of brevity we leave the design of an appropriate Σ_u to the reader as an exercise.

We also note that the static nature of the environment enables us to omit the time index t from the landmark coordinates. Thus, Equation (1.30) can be re-written as follows:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \\ x_1, \\ y_1 \\ \vdots \\ x_N \\ y_N \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \\ x_1, \\ y_1 \\ \vdots \\ x_N \\ y_N \end{pmatrix} + \begin{pmatrix} \Delta t \cos\theta_t & 0 \\ \Delta t \sin\theta_t & 0 \\ 0 & \Delta t \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} + \begin{pmatrix} \varepsilon_{x_t} \\ \varepsilon_{y_t} \\ \varepsilon_{\theta_t} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \tag{1.31}$$

Despite the somewhat tedious derivation, we note that the final result is remarkably simple and easy to implement. Modeling robot motion in Kalman filtering involves simple matrices that are easily computed from the robot's momentary pose and the landmark coordinates.

1.2.4 Linearized Landmark Measurement Model

Kalman filters also require a linear perceptual model with additive Gaussian noise. Put differently, perceptions are modeled by a linear equation of the form

$$z_t = C_t a_t + \varepsilon_{z,t} \tag{1.32}$$

where C_t is a matrix which is yet to be specified, a_t is again the augmented state vector comprising robot pose and the landmark coordinates, and $\varepsilon_{z,t}$ is a zero-centered Gaussian noise variable with covariance matrix $\Sigma_{z,t}$. Written as a conditional probability, we have the following linear Gaussian measurement model, where $|z_t|$ is the dimension of the measurement vector.

$$p(z_t|a_t) = \frac{1}{\sqrt{(2\pi)^{|z_t|} \mid \Sigma_{z,t} \mid}} e^{-\frac{1}{2}(C_t a_t - z_t)^T \Sigma_{z,t}^{-1} (C_t a_t - z_t)} \tag{1.33}$$

The derivation of the matrix C_t and the covariance matrix $\Sigma_{z,t}$ is similar to the that of the linear-Gaussian motion model above. However, whereas in the motion case only the robot's pose s_t was a determining factor in the state prediction, here both the pose s_t and the enviornment (and hence the map m) influence the outcome of robot measurements z_t.

The specific nature of C_t and ε_z depends on the robot's sensors. Throughout this chaper, we consider the case where the robot can sense the *range* (distance) and *bearing* (relative angle) of nearby landmarks. This approch is common in Kalman filter mapping, in part because the map itself it specified as a collection of landmarks. In practice, range and bearing might be obtained by analyzing radar scans or camera images for known structural features of the environment. An equally common assumption is that the robot can uniquely identify individual landmarks, that is, it never confuses one landmark with another. For now, we will adopt this assumption, and postpone any discussion of the *data association problem* that naturally arises in many implementations.

The definition of the measurement model requires care, since the number of landmarks obeserved at any point in time is variable. Due to occlusion and similar effects, the robot may only be able to perceive a subset of all landmarks. Sometimes, it might not even succeed in detecting a landmark at all. This cases the measurement vector to be of variable length. The common approach in the literature is to deal with this variability by resorting to a measurement model of observing a single landmark at-a-time. Multiple landmark sigthings are represented by a sequence of individual single-landmark measurements. This approach enables us to reduce the perceptual model to one that accommodates exactly one landmark observation at-a-time, without sacrificing the ability to process variable-size observation vectors.

Let n_t be the index of the landmark observed at time t. For our landmark-observing robot, we measurements are of the type

$$z_t = \begin{pmatrix} r_t \\ \phi_t \end{pmatrix} \tag{1.34}$$

Here r_t is the Euclidean distance to the landmark n_t, and ϕ_t is the bearing of the landmark relative to the robot's heading direction. In the noise-free measurement case, the values of r_t and ϕ_t are generated using the straightforward laws of geometry:

$$\begin{pmatrix} r_t \\ \phi_t \end{pmatrix} = \begin{pmatrix} \sqrt{(x_{n_t} - x_t)^2 + (y_{n_t} - y_t)^2} \\ \arctan \frac{y_{n_t} - y_t}{x_{n_t} - x_t} - \theta_t \end{pmatrix} \tag{1.35}$$

where $\langle x_t, y_t, \theta_t \rangle$ is the robot's pose at time t, and $\langle x_{n_t}, y_{n_t} \rangle$ are the coordinates of the n_t-th landmark. In practice, measurements are noisy. This noise is conveninently modeled using independent Gaussian noise variables $\varepsilon_{r,t}$ and $\varepsilon_{\phi,t}$:

$$\begin{pmatrix} r_t \\ \phi_t \end{pmatrix} = \begin{pmatrix} \sqrt{(x_{n_t} - x_t)^2 + (y_{n_t} - y_t)^2} \\ \arctan \frac{y_{n_t} - y_t}{x_{n_t} - x_t} - \theta_t \end{pmatrix} + \underbrace{\begin{pmatrix} \varepsilon_{r,t} \\ \varepsilon_{\phi,t} \end{pmatrix}}_{\varepsilon_{z,t}} \tag{1.36}$$

The variables $\varepsilon_{r,t}$ and $\varepsilon_{\phi,t}$ are Gaussian noise variables for range and bearing, respectively. Their mean is zero, and their variances will be denoted $\Sigma_{r,t}$ and $\Sigma_{\phi,t}$. Notice that our formulation does not allow for noise in the identity of an observed landmark n_t. As discussed in the introduction to this chapter, such a noise term would inevitably resul in non-Gaussian posteriors, hence is excluded from the consideraiton here. We reiterate, however, that the lack of such a noise term is the most important shortcoming of the Kalman filter approach to learning maps.

Just like the motion model, the measurement model must be linear for Kalman filters to be applicable. More specifically, we have to approximate (1.36) using the matrix C_t and the noise vector $\varepsilon_{z,t}$. The noise vector $\varepsilon_{z,t}$ was already defined in Equation (1.36). Thus, the missing component to the perceptual model is the matrix C_t, which shall constitute a linearized version of the measurement model (1.36).

It will be convenient to decompose the matrix C_t into multiple submatrices, one for the robot pose s_t and N others for the N different landmarks in the map:

$$C_t = \begin{pmatrix} C_{s,t} & C_{1,t} & C_{2,t}, \ldots, & C_{N,t} \end{pmatrix} \tag{1.37}$$

Here $C_{s,t}$ is a matrix of size 2 by 3, which characterizes the effect of the robot's pose s_t on the measurement z_t. Each matrix $C_{n,t}$ for $1 \leq n \leq N$ is of size 2 by 2, characterizing the effect of the n-th landmark coordinates on the measurement. Since we already know that the robot observed landmark n_t at time t, all but one of these matrices are null matrices. This is because only the n_t-th landmark influences the measurement z_t. This gives us

$$C_t = \begin{pmatrix} C_{s,t} & 0 \cdots 0 & C_{n_t,t} & 0 \cdots 0 \end{pmatrix} \tag{1.38}$$

Both of the remaining matrices, $C_{s,t}$ and $C_{n_t,t}$, are obtained by linearizing the perceptual model defined in Equation (**??**). THe rationale behind this linearization is the same as in the case of the motion model, which was discussed

in the previous section. In the case of $C_{s,t}$, a linearized version approximates the perceptual equations by its first derivatives:

$$C_{s,t} = \frac{\partial z_t}{\partial s_t} = \begin{pmatrix} \frac{\partial r_t}{\partial x_t} & \frac{\partial r_t}{\partial y_t} & \frac{\partial r_t}{\partial \theta_t} \\ \frac{\partial \phi_t}{\partial x_t} & \frac{\partial \phi_t}{\partial y_t} & \frac{\partial \phi_t}{\partial \theta_t} \end{pmatrix} \tag{1.39}$$

Differentiating (**??**) gives us the following matrix:

$$C_{s,t} = \begin{pmatrix} \frac{x_t - x_{n_t}}{\sqrt{(x_t - x_{n_t})^2 + (y_t - y_{n_t})^2}} & \frac{y_t - y_{n_t}}{\sqrt{(x_t - x_{n_t})^2 + (y_t - y_{n_t})^2}} & 0 \\ \frac{y_{n_t} - y_t}{(x_t - x_{n_t})^2 + (y_t - y_{n_t})^2} & \frac{x_t - x_{n_t}}{(x_t - x_{n_t})^2 + (y_t - y_{n_t})^2} & -1 \end{pmatrix} \tag{1.40}$$

Similarly, $C_{n_t,t}$ is obtained by computing the first derivative of the sensor measurement z_t with respect to the landmark location $\langle x_{n_t}, y_{n_t} \rangle$ in the map:

$$C_{n_t,t} = \frac{\partial z_t}{\partial \langle x_{n_t}, y_{n_t} \rangle} = \begin{pmatrix} \frac{\partial r_t}{\partial x_{n_t}} & \frac{\partial r_t}{\partial y_{n_t}} \\ \frac{\partial \phi_t}{\partial x_{n_t}} & \frac{\partial \phi_t}{\partial y_{n_t}} \end{pmatrix} \tag{1.41}$$

This is calculated as follows:

$$C_{n_t,t} = \begin{pmatrix} \frac{x_{n_t} - x_t}{\sqrt{(x_t - x_{n_t})^2 + (y_t - y_{n_t})^2}} & \frac{y_{n_t} - y_t}{\sqrt{(x_t - x_{n_t})^2 + (y_t - y_{n_t})^2}} \\ \frac{y_t - y_{n_t}}{(x_t - x_{n_t})^2 + (y_t - y_{n_t})^2} & \frac{x_{n_t} - x_t}{(x_t - x_{n_t})^2 + (y_t - y_{n_t})^2} \end{pmatrix} \tag{1.42}$$

There exists an interesting relation between the matrices $C_{s,t}$ and $C_{n_t,t}$. In particular, the first two columns in $C_{s,t}$ are equivalent to $-C_{n_t,t}$. The reason for this (as)symmetry lies in the fact that measurements carry only relative information, of the location of a landmark relative to the robot's pose. An increase of the robot's x-coordinate has the same effect on a measurement as a decrease of a landmark's x coordinate in the map. Thus, any measurement system that measures relative information must possess this assymetry.

We are now in the position to forumlate the matrix C_t of the linearized perceptual model. C_t is obtained by substituting $C_{s,t}$ and $C_{n_t,t}$ back into Equation (1.37). The result is the approximation to the non-linear perceptual model defined in Equation (**??**).

$$C_t = \begin{pmatrix} \frac{x_t - x_{n_t}}{\sqrt{q}} & \frac{y_t - y_{n_t}}{\sqrt{q}} & 0 & 0 \ldots 0 & \frac{x_{n_t} - x_t}{\sqrt{q}} & \frac{y_{n_t} - y_t}{\sqrt{q}} & 0 \ldots 0 \\ \frac{y_{n_t} - y_t}{q} & \frac{x_t - x_{n_t}}{q} & -1 & 0 \ldots 0 & \frac{y_t - y_{n_t}}{q} & \frac{x_{n_t} - x_t}{q} & 0 \ldots 0 \end{pmatrix} \tag{1.43}$$

with

$$q = (x_t - x_{n_t})^2 + (y_t - y_{n_t})^2 \tag{1.44}$$

Under this approximation, we can now write

$$z_t = C_t a_t + \varepsilon_{z,t} \tag{1.45}$$

Here $\varepsilon_{z,t}$ are the measurement errors at time t with covariance $\Sigma_{z,t}$.

1.3 THE KALMAN FILTER ALGORITHM FOR CONCURRENT MAPPING AND LOCALIZATION

The definition of the linearized motion and measurement model, and in particular the matrices A_t, B_t, C_t, $\Sigma_{u,t}$, and $\Sigma_{z,t}$, make it now feasible to formulate the Kalman filter algorithm for concurrent mapping and localization. The version stated here assumes a fixed number of landmarks N, which can be identified uniquely. Extensions for variable numbers of landmarks will be discussed further below, as will be strategies for handling the data association problem.

The mapping algorihtm is directly obtained by substituting the specific values of A_t, B_t, C_t, $\Sigma_{u,t}$, and $\Sigma_{z,t}$ into the vanilla Kalman filter, as stated in Table **??** on page **??** of this book. Table 1.1 states the Kalman filter algorithm for the concurrent mapping and localization problem. In this algorithm, the posterior over the augmented state at time t is given by the tuple

$$b_t(a_t) = \langle \mu_t, \Sigma_t \rangle \tag{1.46}$$

where μ_t is the mean of a $(2N + 3)$-dimensional Gaussian and Σ_t is its covariance, with N being the number of landmarks in the map. Notice that the linearized motion and measurement model use the estimated state μ as the linearization point, instead of the true augmented state a, as stated in the previous sections. This accounts for the fact that the true state is of course not observable, and all the robot has is its own estimate regarding the true state. It is therefore common practice to use the mode of this Gaussian estimate, μ, for the linearization.

The description of this recursive algorithm would be incomplete without a discussion as to how the estimate of the augmented state is initialized. The

1: **Algorithm Kalman_filter_mapping(μ, Σ, d):**

2: *if d is a perceptual data item $z = (r, \phi)$ corresponding to landmark n then*

3: $q = (\mu_x - \mu_{x_n})^2 + (\mu_y - \mu_{y_n})^2$

4: $C = \begin{pmatrix} \frac{\mu_x - \mu_{x_n}}{\sqrt{q}} & \frac{\mu_y - \mu_{y_n}}{\sqrt{q}} & 0 & 0 \ldots 0 & \frac{\mu_{x_n} - \mu_x}{\sqrt{q}} & \frac{\mu_{y_n} - \mu_y}{\sqrt{q}} & 0 \ldots 0 \\ \frac{\mu_{y_n} - \mu_y}{q} & \frac{\mu_x - \mu_{x_n}}{q} & -1 & 0 \ldots 0 & \frac{\mu_y - \mu_{y_n}}{q} & \frac{\mu_{x_n} - \mu_x}{q} & 0 \ldots 0 \end{pmatrix}$

5: $K = \Sigma C^T (C \Sigma C^T + \Sigma_z)^{-1}$

6: $\mu = \mu + K(o - C\mu)$

7: $\Sigma = (I - KC)\Sigma$

8: *else if d is an action data item $u = (v, \omega)$ then*

9: $B = \begin{pmatrix} \Delta t \cos \mu_\theta & \Delta t \sin \mu_\theta & 0 & 0 \ldots 0 \\ 0 & 0 & \Delta t & 0 \ldots 0 \end{pmatrix}^T$

10: $\mu = \mu + Bu$

11: $\Sigma = \Sigma + \Sigma_u$

12: *return* $\langle \mu, \Sigma \rangle$

Table 1.1 The Kalman filter algorithm for concurrent mapping and localization. It incrementally calculates the posterior over the joint space of robot poses and maps, represented by a multivariate Gaussian distribution.

mean of the Gaussian is initialized by the null vector

$$\mu_0 = \begin{pmatrix} 0 & \cdots & 0 \end{pmatrix}^T \tag{1.47}$$

and the covariance matrix is initialized as follows:

$$\Sigma_0 = \begin{pmatrix} 0 & 0 & 0 & \infty & \ldots & \infty \\ 0 & 0 & 0 & \infty & \ldots & \infty \\ 0 & 0 & 0 & \infty & \ldots & \infty \\ \infty & \infty & \infty & \infty & \ldots & \infty \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \infty & \infty & \ldots & \infty \end{pmatrix} \tag{1.48}$$

The initial covariance Σ_0 along with the initial mean estimate μ_0 specify that the initial pose is $\langle x = 0, y = 0, \theta = 0 \rangle$ with absolute certainty. This establishes the boundary condition for mapping: Since all estimates are in global

coordinates yet all sensor measurements are relative, one of the poses must be specified in advance in global coordinates. The initial covariance matrix also specifies that the landmark locations are entirely unknown. In other words, the covariance in the landmark estimate in ∞, which implements a uniform prior over landmark locations. This is plausible, as the initial location of the landmarks is unknown. In practice, the initialization of landmark covariance estimates with ∞ is a bit 'academic,' as estimates for never-seen landmarks have not to be computed. The actual size of the estimate, thus, depends on the number of landmarks that have been sighted.

Finally, let us briefly consider the computational costs of the algorithm in Table 1.1. Our algorithm requires $O(N^2)$ space to store the belief space, with N being the number of landmarks in the map. For large number of landmarks, updating the belief takes $O(N^2)$ time. This makes this algorithm computationally cumbersome if the number of landmarks is large (e.g., larger than 100). One might be tempted to approximate the posterior $b_t(a_t)$ by omitting certain off-diagonal elements in the covariance matrix Σ_t. However, the off-diagonal elements are very important, as they capture the correlations between the uncertainty in the robot's pose and the uncertainty in the map. In fact, as time goes to infinity, the off-diagonal elements in the covariance matrix can be shown to converge to 1, making the pose estimation and the landmark estimation fully correlated (for the linear case). Today, it is commonly believed that the full covariance matrix must be carried along in the calculation, at least if one seeks to maintain a single, monolithic global map.

1.3.1 Example

Kalman filter based mapping has been applied successfully to a large range of navigation problems, involving airborne, underwater, indoor, and various other vehicles. Figure 1.1 shows an example result obtained using the underwater robot Oberon, developed at the University of Sydney and shown in Figure 1.2 (courtesy of Stefan Williams and Hugh Durrant-Whyte). This vehicle is equipped with a pencil sonar, a sonar that can scan at very high resolutions an detect obstacles up to 50 meters away. To facilitate the mapping problem, researchers have deposited long, small vertical objects in the water, which can be extracted from the sonar scans with relative ease. In this specific experiment, there is a row of such objects, spaced approximately 10 meters apart. In addition, a more distant cliff offers addiitonal point features that can be detected using the pencil sonar. In the experiment shown in Figure 1.1, the robot moves by these landmarks, then turns around and moves back. While

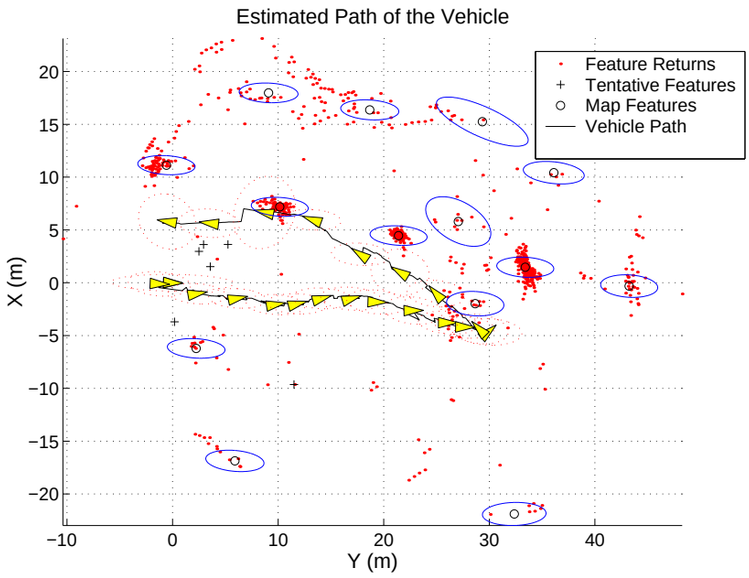

Figure 1.1 Example of Kalman filter estimation of the map and the vehicle pose. Courtesy of the Australian Centre for Field Robotics, University of Sydney, Australia.

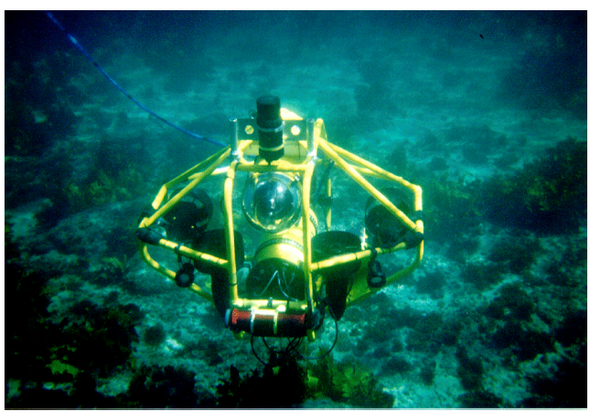

Figure 1.2 Underwater vehicle Oberon, developed at the University of Sydney. Image courtesy of the Australian Centre for Field Robotics, University of Sydney, Australia.

doing so, it measures and integrates detected landmarks into its map. using the Kalman filter approach described in this chapter.

The map shown in Figure 1.1 shows the robot's path, marked by the triangles connected by a line. Around each triangle one can see an ellipse, which corresponds to the covariance matrix of the Kalman filter estimate projected into the robot's x-y position. The ellipse shows the variance; the larger it is, the less certain the robot is about its current pose. Various small dots in Figure 1.1 show landmark sightings, obtained by searching the sonar scan for small and highly reflective objects. The majority of these sightings is rejected, using a mechanism described further below in this chapter. However, same are believed to correspond to a landmark and added to the map. At the end of the run, the robot has classified 14 such objects as landmarks, each of which is plotted with the projected uncertainty ellipse in Figure 1.1. These landmarks include the artificial landmarks put out by the researchers, but they also include various other terrain features in the vicinity of the robot. The residual pose uncertainty is small. In this example, the robot successfully finds and maps all of the artificial landmarks in this highly noisy domain.

1.4 MAXIMUM LIKELIHOOD METHODS FOR DATA ASSOCIATION

As discussed in the introduction to this chapter, a key problem of the Kalman filtering approach arises from the data association problem. Data association is the problem of establishing correspondence between different measurements of the potentially same environment feature. The basic Kalman filter algorithm described thus far assumed that the identity of landmarks can be identified with absolute certainty. More specifically, if the robot believes to observe the i-th landmark, it is impossible that it mistook the j-th landmark for the i-th one with $i \neq j$. Hence, it requires that the data association problem is solved. For this reason, early implementation of this algorithm often relied on artificial landmarks, such as bar codes, which could be uniquely identified (e.g., a different bar code for each landmark).

The inability of Kalman filtering to cope with the data association problem arises from the very nature of the Gaussian representation. Suppose we observe a landmark, but we are not quite sure if it was landmark i or landmark j, for $i \neq j$. Then the observation z_t and the matrix $C_t y_t$ is non-zero either for the i-th landmark component, or the j-th. Mixing both in proportion to the probability

that either i or j is the oberved landmark is, unfortunately, incorrect, as the estimation problem is non-convex. Instead, the posterior over the augmented state vector becomes bi-modal, where each mode corresponds to one of the hypotheses. The basic Kalman filter cannot represent multi-modal posterior distributions, which is one of its primary deficiencies. For this reason, our algorithm must require that landmarks are uniquely identifiable.

In practice, however, one usually faces a data association problem. For example, in indoor navigation one often uses features such as T-junctions or doors as landmarks. If the environment possesses more than one T-junction or more than one door, there is the danger of accidentally confusing one landmark with another. Similar problems arise in virtually any application domain. For example, one might use trees as landmarks when mapping outdoor environments, or vertical structures when estimating underwater maps. In each case, landmarks are ambiguous and one runs danger of confusing one landmark with another. If this happens, Kalman filters may fail catastrophically.

The common approach to deal with the data association problem is to use a maximum likelihood estimator. This estimator determines the most likely data association, which is then assumed to be ground truth. To calculate the most likely data association requires us to calculate the probability $p(n|d_{1\ldots t})$ for any possible landmark number n, with $1 \leq n \leq N$, assuming that the most recent data item d_t as a perceptual item z_t. This conditional probability tells us how probable it is that the most recent data item, z_t was caused by landmark n in our map, for any value of n.

It turns out $p(n|d_{1\ldots t})$ can easily be calculated in the Kalman filter regime. Bayes rule suggests that

$$\begin{aligned} p(n|d_{1\ldots t}) &= p(n|d_{1\ldots t-1}, z_t) \\ \eta\, p(z_t|n, d_{1\ldots t-1})\, p(n|d_{1\ldots t-1}) & \end{aligned} \tag{1.49}$$

where η is the familiar normalization constant. Since in the absence of a sensor measurement at time t, one may reasonably assume that all values of n are equally likely, we can safaly assume that $p(n|d_{1\ldots t-1})$ is uniformly distirbted over all n, hence we have

$$\begin{aligned} &= \eta\, p(z_t|n, d_{1\ldots t-1})\, p(n) \\ &= \eta'\, p(z_t|n, d_{1\ldots t-1}) \end{aligned} \tag{1.50}$$

The term $p(z_t|n, d_{1\ldots t-1})$ describes the probability of meassuring z_t conditioned on all past data. The most convenient way to calculate this probability is to

exploit the posterior $p(a_{t-1}|d_{1\ldots t-1})$ calculated by the Kalman filter. Here we exploit the fact that if d_t is a perceptual data item, the robot does not move and hence $a_t = a_{t-1}$.

$$\begin{aligned} p(z_t|n, d_{1\ldots t-1}) &= \int p(z_t|n, d_{1\ldots t-1}, a_t)\; p(a_t|n, d_{1\ldots t-1})\; da_t \\ &= \int p(z_t|n, d_{1\ldots t-1}, a_{t-1})\; p(a_{t-1}|n, d_{1\ldots t-1})\; da_{t-1} \end{aligned} \tag{1.51}$$

which, due to obvious independences, can be sipliefied as follows:

$$\begin{aligned} &= \int p(z_t|n, a_{t-1})\; p(a_{t-1}|d_{1\ldots t-1})\; da_{t-1} \\ &= \int p(z_t|n, a_{t-1})\; b_{t-1}(a_{t-1})\; da_{t-1} \end{aligned} \tag{1.52}$$

As usual, $b_{t-1}(a_{t-1})$ denotes the posterior belief of the augmented state a_{t-1} given the data leading up to time $t-1$.

In general, integrals of this type are difficult to calculate, due to the high dimensionality of the augmented state variable a_{t-1} (and hence a_t) over which one has to integrate. However, in the context of Kalman filters, this integral finds a nice closed-form solution. In particular, we notice that both probability densities inside the integral, $p(z_t|n, a_t)$ and $p(a_t|d_{1\ldots t})$, are Gaussians. In particular, $p(z_t|n, a_t)$ is a Gaussian with mean $C_t a_t$ and covariane $\Sigma_{u,t}$ (see Equation (1.33). The posterior $p(a_t|d_{1\ldots t})$ is a Gaussian with mean μ_t and covariance Σ_t. Convolving both, as in the integral above, leads to another Gaussian. This Gaussian has the mean

$$C_t \mu_t \tag{1.53}$$

and the covariance

$$C_t \Sigma_t C_t^T + \Sigma_{z,t} \tag{1.54}$$

The derivation of this well-known fact follows directly from the Appendix A, where the convolution is solved in the space of Fourier transforms. Written as a probability, we thus have

$$p(n|d_{1\ldots t}) = \eta' \frac{1}{\sqrt{(2\pi)^2 \mid C_t \Sigma_t C_t^T + \Sigma_{z,t} \mid}}\; e^{-\frac{1}{2}(C_t \mu_t - z_t)^T [C_t \Sigma_t C_t^T + \Sigma_{z,t}]^{-1} (C_t \mu_t - z_t)} \tag{1.55}$$

Here the normalization constant η' is calculated over all possible indices n. As a result, we can calculate the probability $p(n|z_t, d_{1\ldots t})$ that the t-the sensor measurment, z_t, was caused by the n-th landmark.

The maximum likelihood approach for data association assumes that the index of the most likely landmark under this likelihood function is the correct one. Put differently, n is obtained via the following maximizaiton:

$$n_t = \operatorname*{argmax}_n p(n|d_{1\ldots t}) \tag{1.56}$$

This leads to a practical algorithm for calculating the most likly data association given an augmented state estimate $b_{t-1}(a_{t-1})$ and a sensor measurement z_t.

Unfortunately, this approach to data association has a clear limitation. This limitation arises from the fact that the maximum likelihood approach deviates from the idea of full posterior estimation for concurrent mapping and estimation. Instead of maintaining a joint posterior over augmented states and data associations, it reduces the data association problem to a deterministic determination, which is treated as if the maximum likelihood association was always correct. This limitation makes Kalman filtering brittle with regards to landmark confusion, which in turn can lead to wrong results.

In practice, researchers often remedy the probelm by choosing one of the following two methods, both of which reduce the chances of confusing landmarks:

- **Spatial arrangement.** The further apart landmarks are, the smaller the chance to accidentally confuse them. It is therefore common practice to choose landmarks that are sufficiently far away from each other to that the probability of confusing one with another is negligible. This introduces an interesting trade-off: there might not be too many landmarks, since the danger of confusing them increases with their density. There might also not to be too few, since the fewer landmarks there are, the more difficult is it to localize the robot and build a map. Little is currently known about the optimal density of landmarks, and researchers often use intuition when selecting specific landmarks.
- **Signatures.** Often, natural landmarks are perceptually distinct. For example, doors might possess different colors, or corridors might have different widths. Perceptual features of landmarks are usually referred to as *signatures*. Signatures are easily factored into the map estimation algorithm, and can be used to help discriminate different landmarks.

With these two strategies, the Kalman filter approach can be applied to a wide range of practical mapping problems, even if landmarks are not identifiable. With high probability the resulting estimate resembles the true posterior,

which carries information about the map, the robot pose, the corresponding uncertainties, and the dependence between robot and map estimates.

1.5 OUTLIER REMOVAL

A similar problem arises from spurious measurements, which are measurements that are caused by something else than the landmarks in the map. This is another facet of the data association problem: Does a landmark observation correspond to any of the landmarks in the map, or has it been caused by something else (e.g., noise)? In Gaussian estimation, the effect of a measurement on the augmented state estimate μ depends on the $P(z_t|d)$. Measurements that look very unlikely under this distribution actually have the strongest effect. The problem with incorporating spurious measurements, thus, is that they can severely corrupt the estimation. In practice, it is therefore important to identity and reject such measurements.

Unfortunately, we cannot apply the maximum likelihood algorithm above for the outlier removal problem, since the Kalman filter does not model the effect that causes the outlier. The common approach to this problem is to reject observations that surprise too much. The probability of a measurement is given by (**??**). The logarithm of the exponential in (**??**) multiplied by -2 is the distance function

$$(z_t - C_t\mu_t)^T \left[C_t\Sigma_t C_{tT} + \Sigma_z\right]^{-1} (z_t - C_t\mu_t) \tag{1.57}$$

This type of weighted quadratic distance is known as Mahalanobis distance. The level set of vectors z_t for which this distance is constant is an ellipse. As a simple heuristic, measurements z_t are simply discard if their Mahalanobis distance exceeds a certain threshold. In practice, it is usually good to be conservative, and fix the threshold to be a value such as 1, that is, reject all measurement that fall outside a single variance of the expected measurement. With such a tight threshold, a good number of "good" measurements will be discarded accidentally, but this is usually less damaging than updating the Kalman filter with spurious measurements.

Table 1.2 shows the Kalman filter algorithm augmented by maximum likelihood data association and outlier removal, as discussed in this section. Practical implementations of Kalman filter mapping typically rely on this, more robust approach, instead of the one shown in Table 1.1.

1: **Algorithm robust_Kalman_filter_mapping**(μ, Σ, d)**:**

2: *if* d *is a perceptual data item* $o = (r, \phi)$ *then*
3: mindist $= \infty$
4: *for all landmarks* i *do*
5: $q = (\mu_x - \mu_{x_i})^2 + (\mu_y - \mu_{y_i})^2$
6: $C' = \begin{pmatrix} \frac{\mu_x - \mu_{x_i}}{\sqrt{q}} & \frac{\mu_y - \mu_{y_i}}{\sqrt{q}} & 0 & 0 \ldots 0 & \frac{\mu_{x_i} - \mu_x}{\sqrt{q}} & \frac{\mu_{y_i} - \mu_y}{\sqrt{q}} & 0 \ldots 0 \\ \frac{\mu_{y_i} - \mu_y}{q} & \frac{\mu_x - \mu_{x_i}}{q} & -1 & 0 \ldots 0 & \frac{\mu_y - \mu_{y_i}}{q} & \frac{\mu_{x_i} - \mu_x}{q} & 0 \ldots 0 \end{pmatrix}$
7: dist $= (o - C'\mu)^T \left[C\Sigma C^T + \Sigma_z\right]^{-1} (o - C'\mu)$
8: *if* dist $<$ mindist
9: mindist $=$ dist
10: $C = C'$

11: *if* mindist ≤ 1
12: $K = \Sigma C^T (C\Sigma C^T + \Sigma_z)^{-1}$
13: $\mu = \mu + K(o - C\mu)$
14: $\Sigma = (I - KC)\Sigma$

15: *else if* d *is an action data item* $a = (v, \omega)$ *then*
16: $B = \begin{pmatrix} \Delta t \cos \mu_\theta & \Delta t \sin \mu_\theta & 0 & 0 \ldots 0 \\ 0 & 0 & \Delta t & 0 \ldots 0 \end{pmatrix}^T$
17: $\mu = \mu + Ba$
18: $\Sigma = \Sigma + \Sigma_{u,t}$

19: *return* $\langle \mu, \Sigma \rangle$

Table 1.2 This version of the Kalman fiter algorithm can generate maps with ambigiuous landmakrs and is robust to spurious measurements. The threshold for outlier removal is set to the variance (line 11).

1.6 DYNAMIC FEATURE SETS

Another common feature of Kalman filter mapping algorithms is to dynamically change the number of landmarks. Thus far, we assumed that the number of landmarks (or an upper bound thereof) is known a priori. In practice, however, the number of landmarks is typically not known, making it desirable to find algorithms that can dynamically change the number of landmarks in the map.

Fortunately, adding and removing landmarks is relatively straightforward in the Kalman filter approach. When adding a landmark, the augmented state estimate μ is augmented by two additional values, corresponding to the coordinate estimates of the new landmark. Similarly, the covariance matrix Σ is augmented by two more rows and columns, which specify the correlations between the corrdinates of the new landmark, and all previous landmarks along with the robot's pose. How should we initialize those new values? The answer is easy: initialize the covariance elements by ∞, to indicate that we have no a priori information about the location of the landmark. The location estimate in μ is then irrelevant and can be initialized by any value (e.g., $\langle 0, 0 \rangle$). Typically, landmarks are added when they are observed; that is, without loss of generality let us assume that the most recent observation z_t corresponds to a new landmark. Incorporating this obseervation via the Kalman filter update (Lines 3 throught 6 in Table 1.1, page 20) leads to a more sensible estimate of the landmark location and the corresponding covariance terms. Removing a landmark is even simpler. One simply has to remove the correspinding entries from the vector μ and the matrix Σ. This is the same as projecting the multivariate Gaussian into the subspace that lacks the landmark that is being removed.

The ability to add and remove landmarks from the map on-line raises the question as to *when* to perform these operations. At first glance, one might be tempted to add every landmark measurement rejected by the Mahalanobis distance-based outlier rejection strategy described above (possibly with a larger threshold than 1). However, in highly noisy domains this can have devastating effects, as the number of landmarks in the map can grow unreasonably fast.

A common approach is to maintain an *candidate list* of potential new landmarks. There exists a variety of mostly ad hoc ways to maintain candidate lists in the literature. Here we outline one such potential scheme. When an unknown landmark is observed, its location estimate, along with its covariance, is added to the candidate list. Whenever a landmark is seen in the vicinity of the candidate, a counter is incremented; when the robot does not observe

anything while being in sensor range of the candidate landmark, the counter is decremented. If the counter surpasses an acceptance threshold (e.g., three sightings), it is added to the map. Similarly, counters attached to landmarks already in the map can be used to remove landmarks from the map. The removal of landmarks is important, as any adding scheme might accidentially add a non-existent landmark.

1.7 SUMMARY

This section described algorithms for concurrent mapping and localization based on Kalman filters. In particular:

- We discussed the basic assumptions that are being made when applying Kalman filters to mapping: The existence of point-features (landmarks), the linearity of the motion model and the perceptual model, the Gaussian noise assumption in robot motion and perception, and the inabiulity to process negative measurements. From those, the Gaussian noise assumption in perception is most critical, as it implies that landmarks can be identified uniquely.
- We showed how to linearize robot motion and perception, leading to a practical mapping algorithm.
- We then discussed how to use a maximum likelihood estimator to cope with the data association problem that arises when landmarks cannot be identified uniquely.
- We also showed how to devise a statistical test for rejecting outliers, which is crucial for the working of this method in noisy domains.
- We finally discussed how to use candidate lists to dynamically grow and shrink the number of landmarks in the map. This extension is important if the total number of landmarks is not known in advance.

Kalman filter approaches are currently the only mapping algorithms that can tractably compute the full posterior over maps and poses, using an incremental (recursive) update equation. The advantages of a full posterior estimate are intriguing: At any point in time, one can extract the map and the robot pose along with the uncertainty, and one can also extract the cross-dependencies

between the robot and the map estimate. Moreover, under the restrictive assumptions above the Kalman filter-based mapping algorithm can be proven to converge to the correct map.

Kalman filters gain this advantage thorugh a series of restrictive assumptions, that limits their applicability in practice. The most important restriction is the reliance on features, in a way that facilitates their disambiguation. As a result, Kalman filter implementaitons typically throw away the vast majority of sensor data, using only a small subset (sparse features) for mapping. In many environment, the existence of appropriate features is questionable, and the reduction of data through feature extraction makes the mapping problem much harder. In comparison, the mapping algorithms discussed in the previous and the next chapter are able to operate on raw data, and they can cope with ambiguous sensor measurements, thereby solving the data association problem (subject to other assumptions).

1.8 BIBLIOGRAPHICAL REMARKS

Place them here!

1.9 PROJECTS

1. Develop an incremental algorithm for posterior estimation of poses and maps (with known data association) that does not rely on linearizing the motion model and the perceptual model. Our suggesion: Replace the Kalman filter by particle filters.

2. The basic Kalman filter approach is unable to cope with the data association problem in a sound statistical way. Develop an algorithm (and a statistical framework) for posterior estimation with unknown data association, and characterize its advantages and disadvantages. We suggest to use a mixture of Gaussians representation.

3. Based on the previous problem, develop an approximate method for posterior estimation with unknown data association, where the time needed for each incremental update step does not grow over time (assuming a fixed number of landmarks).

4. Develop a Kalman filter algorithm which uses local occupancy grid maps as its basic components, instead of landmarks. Among other things, problems that have to be solved are how to relate local grids to each other, and how to deal with the ever-growing number of local grids.

5. Develop and algorithm that learns its own features for Kalman filter mapping. There could be two different variants: One which chooses features so as to minimize the uncertainty (entropy) in the posterior, another which is given access to the correct map and seeks to maximize the probability of the correct map under the posterior.