feature in the map:

$$F_j = \begin{pmatrix} 0\cdots 0 & 1\ 0\ 0 & 0\cdots 0 \\ 0\cdots 0 & 0\ 1\ 0 & 0\cdots 0 \\ 0\cdots 0 & \underbrace{0\ 0\ 1}_{j-\text{th feature}} & 0\cdots 0 \end{pmatrix} \tag{11.34}$$

This insight makes it possible to decompose the implement Equations (11.31) and (11.32) into a sequential update:

$$\tilde{\Omega} = \Omega_{x_{0:t},x_{0:t}} - \sum_j \Omega_{x_{0:t},j}\ \Omega_{j,j}^{-1}\ \Omega_{j,x_{0:t}} \tag{11.35}$$

$$\tilde{\xi} = \xi_{x_{0:t}} - \sum_j \Omega_{x_{0:t},j}\ \Omega_{j,j}^{-1}\ \xi_j \tag{11.36}$$

The matrix $\Omega_{x_{0:t},j}$ is non-zero only for elements in $\tau(j)$, the set of poses at which feature j was observed. This essentially proves the correctness of the reduction algorithm **GraphSLAM_reduce** in Table 11.3. The operation performed on Ω in this algorithm can be thought of as the variable elimination algorithm for matrix inversion, applied to the feature variables but not the robot pose variables.

11.4.6 Recovering the Path and the Map

The algorithm **GraphSLAM_solve** in Table 11.4 calculates the mean and variance of the Gaussian $\mathcal{N}(\tilde{\xi}, \tilde{\Omega})$, using the standard equations, see Equations (3.72) and (3.73) on page 72:

$$\tilde{\Sigma} = \tilde{\Omega}^{-1} \tag{11.37}$$

$$\tilde{\mu} = \tilde{\Sigma}\,\tilde{\xi} \tag{11.38}$$

In particular, this operation provides us with the mean of the posterior on the robot path; it does not give us the locations of the features in the map.

It remains to recover the second factor of Equation (11.26):

$$p(m \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t}) \tag{11.39}$$

The *conditioning lemma*, stated and proved in Table 11.7, shows that this probability distribution is Gaussian with the parameters

$$\Sigma_m = \Omega_{m,m}^{-1} \tag{11.40}$$

$$\mu_m = \Sigma_m(\xi_m - \Omega_{m,x_{0:t}}\,\tilde{\mu}) \tag{11.41}$$

Here ξ_m and $\Omega_{m,m}$ are the subvector of ξ, and the submatrix of Ω, respectively, restricted to the map variables. The matrix $\Omega_{m,x_{0:t}}$ is the off-diagonal submatrix of Ω that connects the robot path to the map. As noted before, $\Omega_{m,m}$ is block-diagonal, hence we can decompose

$$p(m \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t}) = \prod_j p(m_j \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t}) \tag{11.42}$$

where each $p(m_j \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$ is distributed according to

$$\Sigma_j = \Omega_{j,j}^{-1} \tag{11.43}$$

$$\mu_j = \Sigma_j(\xi_j - \Omega_{j,x_{0:t}}\tilde{\mu}) = \Sigma_j(\xi_j - \Omega_{j,\tau(j)}\tilde{\mu}_{\tau(j)}) \tag{11.44}$$

The last transformation exploited the fact that the submatrix $\Omega_{j,x_{0:t}}$ is zero except for those pose variables $\tau(j)$ from which the j-th feature was observed.

It is important to notice that this is a Gaussian $p(m \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$ conditioned on the true path $x_{0:t}$. In practice, we do not know the path, hence one might want to know the posterior $p(m \mid z_{1:t}, u_{1:t}, c_{1:t})$ without the path in the conditioning set. This Gaussian cannot be factored in the moments parameterization, as locations of different features are correlated through the uncertainty over the robot pose. For this reason, **GraphSLAM_solve** returns the mean estimate of the posterior but only the covariance over the robot path. Luckily, we never need the full Gaussian in moments representation—which would involve a fully populated covariance matrix of massive dimensions—as all essential questions pertaining to the SLAM problem can be answered at least in approximation without knowledge of Σ.

11.5 Data Association in GraphSLAM

Data association in GraphSLAM is realized through correspondence variables, just as in EKF SLAM. GraphSLAM searches for a single best correspondence vector, instead of calculating an entire distribution over correspondences. Thus, finding a correspondence vector is a search problem. However, it shall prove convenient to define correspondences slightly differently in GraphSLAM than before: correspondences are defined over pairs of features in the map, rather than associations of measurements to features. Specifically, we say $c(j,k) = 1$ if m_j and m_k correspond to the same physical feature in the world. Otherwise, $c(j,k) = 0$. This feature-correspondence is in fact logically equivalent to the correspondence defined in the previous section, but it simplifies the statement of the basic algorithm.