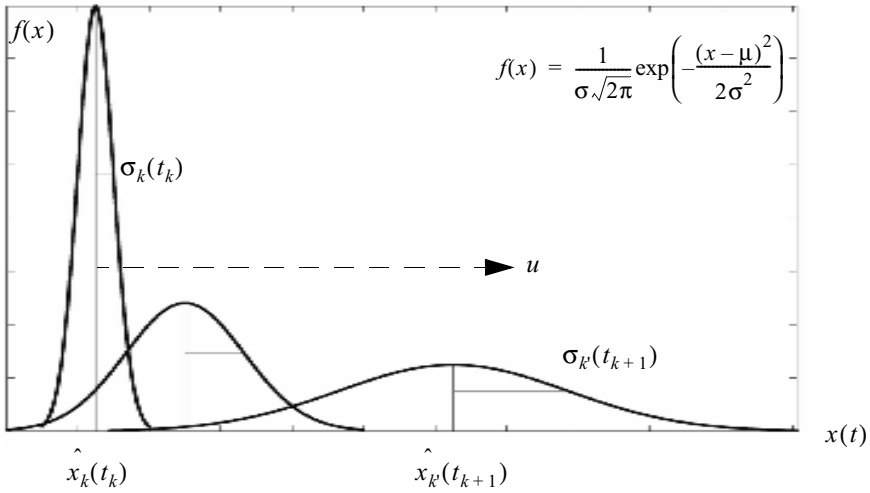

Figure 5.27
Propagation of probability density of a moving robot [106].

The optimal estimate at time $k+1$ is given by the last estimate at k and the estimate of the robot motion including the estimated movement errors.

By extending the above equations to the vector case and allowing time-varying parameters in the system and a description of noise, we can derive the Kalman filter localization algorithm.

5.6.3.2 Application to mobile robots: Kalman filter localization

The Kalman filter is an optimal and efficient sensor fusion technique. Application of the Kalman filter to localization requires posing the robot localization problem as a sensor fusion problem. Recall that the basic probabilistic update of robot belief state can be segmented into two phases, *perception update* and *action update,* as specified by equations (5.21) and (5.22).

The key difference between the Kalman filter approach and our earlier Markov localization approach lies in the perception update process. In Markov localization, the entire perception, that is, the robot's set of instantaneous sensor measurements, is used to update each possible robot position in the belief state individually using the Bayes formula. In some cases, the perception is abstract, having been produced by a feature extraction mechanism, as in Dervish. In other cases, as with Rhino, the perception consists of raw sensor readings.

By contrast, perception update using a Kalman filter is a multistep process. The robot's total sensory input is treated not as a monolithic whole but as a set of extracted features that

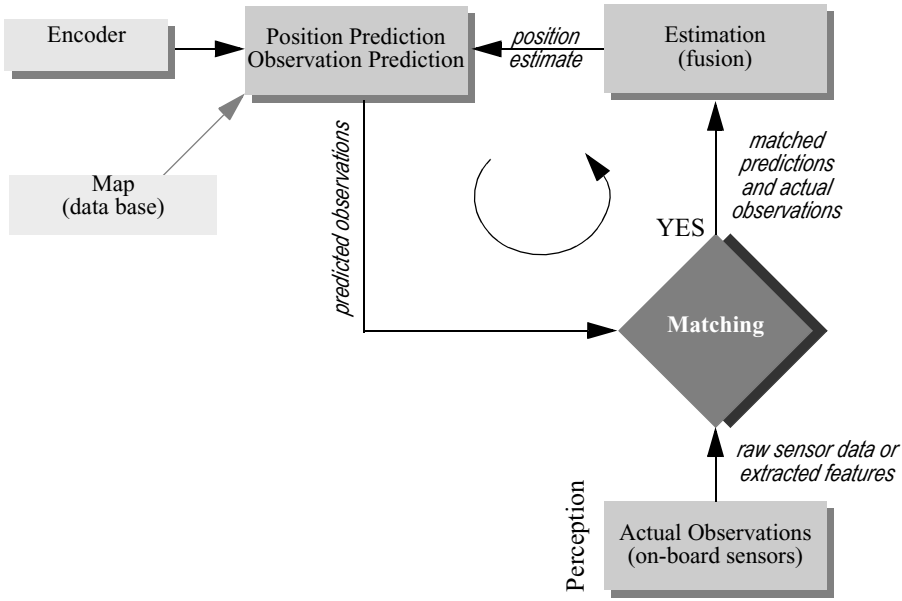

Figure 5.28
Schematic for Kalman filter mobile robot localization (see [23]).

each relate to objects in the environment. Given a set of possible features, the Kalman filter is used to fuse the distance estimate from each feature to a matching object in the map. Instead of carrying out this matching process for many possible robot locations individually as in the Markov approach, the Kalman filter accomplishes the same probabilistic update by treating the whole, unimodal, and Gaussian belief state at once. Figure 5.28 depicts the particular schematic for Kalman filter localization.

The first step is action update or *position prediction*, the straightforward application of a Gaussian error motion model to the robot's measured encoder travel. The robot then collects actual sensor data and extracts appropriate features (e.g., lines, doors, or even the value of a specific sensor) in the *observation* step. At the same time, based on its predicted position in the map, the robot generates a *measurement prediction* which identifies the features that the robot expects to find and the positions of those features. In *matching* the robot identifies the best pairings between the features actually extracted during observation and the expected features due to measurement prediction. Finally, the Kalman filter can fuse the information provided by all of these matches to update the robot belief state in *estimation*.

In the following sections these five steps are described in greater detail. The presentation is based on the work of Leonard and Durrant-Whyte [23, pp. 61–65].

1. Robot position prediction. The robot's position at timestep $k+1$ is predicted based on its old location (at timestep k) and its movement due to the control input $u(k)$:

$$\hat{p}(k+1|k) = f(\hat{p}(k|k), u(k)) \tag{5.46}$$

For the differential-drive robot, $\hat{p}(k+1|k) = p'$ is derived in equations (5.6) and (5.7) respectively.

Knowing the plant and error model, we can also compute the variance $\Sigma_p(k+1|k)$ associated with this prediction [see equation. (5.9), section 5.2.4]:

$$\Sigma_p(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T \tag{5.47}$$

This allows us to predict the robot's position and its uncertainty after a movement specified by the control input $u(k)$. Note that the belief state is assumed to be Gaussian, and so we can characterize the belief state with just the two parameters $\hat{p}(k+1|k)$ and $\Sigma_p(k+1|k)$.

2. Observation. The second step it to obtain sensor measurements $Z(k+1)$ from the robot at time $k+1$. In this presentation, we assume that the observation is the result of a feature extraction process executed on the raw sensor data. Therefore, the observation consists of a set n_0 of single observations $z_j(k+1)$ extracted from various sensors. Formally, each single observation can represent an extracted feature such as a line or door, or even a single, raw sensor value.

The parameters of the features are usually specified in the sensor frame and therefore in a local reference frame of the robot. However, for matching we need to represent the observations and measurement predictions in the same frame $\{S\}$. In our presentation we will transform the measurement predictions from the global coordinate frame to the sensor frame $\{S\}$. This transformation is specified in the function h_i discussed in the next paragraph.

3. Measurement prediction. We use the predicted robot position $\hat{p}(k+1|k)$ and the map $M(k)$ to generate multiple predicted feature observations z_t. Each predicted feature has its position transformed into the sensor frame:

$$\hat{z_i}(k+1) = h_i(z_t, \hat{p}(k+1|k)) \tag{5.48}$$

We can define the measurement prediction as the set containing all n_t predicted feature observations:

$$\hat{Z}(k+1) = \{\hat{z}_i(k+1) | (1 \le i \le n_t)\} \tag{5.49}$$

The predicted state estimate $\hat{p}(k+1|k)$ is used to compute the measurement Jacobian ∇h_i for each prediction. As you will see in the example below, the function h_i is mainly a coordinate transformation between the world frame and the sensor frame.

4. Matching. At this point we have a set of actual, single observations, which are features in sensor space, and we also have a set of predicted features, also positioned in sensor space. The matching step has the purpose of identifying all of the single observations that match specific predicted features well enough to be used during the estimation process. In other words, we will, for a subset of the observations and a subset of the predicted features, find pairings that intuitively say "this observation is the robot's measurement of this predicted feature based on the map."

Formally, the goal of the matching procedure is to produce an assignment from observations $z_j(k+1)$ to the targets z_t (stored in the map). For each measurement prediction for which a corresponding observation is found we calculate the innovation $v_{ij}(k+1)$. *Innovation* is a measure of the difference between the predicted and observed measurements:

$$\begin{aligned} v_{ij}(k+1) &= [z_j(k+1) - \hat{z}_i(k+1)] \\ &= [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))] \end{aligned} \tag{5.50}$$

The innovation covariance $\Sigma_{IN,ij}(k+1)$ can be found by applying the error propagation law [section 4.2.2, equation (4.60)]:

$$\Sigma_{IN,ij}(k+1) = \nabla h_i \cdot \Sigma_p(k+1|k) \cdot \nabla h_i^T + \Sigma_{R,i}(k+1) \tag{5.51}$$

where $\Sigma_{R,i}(k+1)$ represents the covariance (noise) of the measurement $z_i(k+1)$.

To determine the validity of the correspondence between measurement prediction and observation, a *validation gate* has to be specified. A possible definition of the validation gate is the Mahalanobis distance:

$$v_{ij}^T(k+1) \cdot \Sigma_{IN,ij}^{-1}(k+1) \cdot v_{ij}(k+1) \le g^2 \tag{5.52}$$

However, dependent on the application, the sensors, and the environment models, more sophisticated validation gates might be employed.

The validation equation is used to test observation $z_j(k+1)$ for membership in the validation gate for each predicted measurement. When a single observation falls in the validation gate, we get a successful match. If one observation falls in multiple validation gates,

the best matching candidate is selected or multiple hypotheses are tracked. Observations that do not fall in the validation gate are simply ignored for localization. Such observations could have resulted from objects not in the map, such as new objects (e.g., someone places a large box in the hallway) or transient objects (e.g., humans standing next to the robot may form a line feature). One approach is to take advantage of such unmatched observations to populate the robot's map.

5. Estimation: applying the Kalman filter. Next we compute the best estimate $\hat{p}(k+1|k+1)$ of the robot's position based on the position prediction and all the observations at time $k+1$. To do this position update, we first stack the validated observations $z_j(k+1)$ into a single vector to form $z(k+1)$ and designate the composite innovation $v(k+1)$. Then we stack the measurement Jacobians ∇h_i for each validated measurement together to form the composite Jacobian ∇h and the measurement error (noise) vector $\Sigma_R(k+1) = diag[\Sigma_{R,i}(k+1)]$. We can then compute the composite innovation covariance $\Sigma_{IN}(k+1)$ according to equation (5.51) and by utilizing the well-known result [3] that the Kalman gain can be written as

$$K(k+1) = \Sigma_p(k+1|k) \cdot \nabla h^T \cdot \Sigma_{IN}^{-1}(k+1) \tag{5.53}$$

we can update the robot's position estimate

$$\hat{p}(k+1|k+1) = \hat{p}(k+1|k) + K(k+1) \cdot v(k+1) \tag{5.54}$$

with the associated variance

$$\Sigma_p(k+1|k+1) = \Sigma_p(k+1|k) - K(k+1) \cdot \Sigma_{IN}(k+1) \cdot K^T(k+1) \tag{5.55}$$

For the 1D case and with $h_i(z_t, \hat{p}(k+1|k)) = z_t$ we can show that this formula corresponds to the 1D case derived earlier

Equation (5.53) is simplified to

$$K(k+1) = \frac{\sigma_p^2(k+1|k)}{\sigma_{IN}^2(k+1)} = \frac{\sigma_p^2(k+1|k)}{\sigma_p^2(k+1|k) + \sigma_R^2(k+1)} \tag{5.56}$$

corresponding to equation (5.45), and equation (5.54) simplifies to

$$\begin{aligned}\hat{p}(k+1|k+1) &= \hat{p}(k+1|k) + K(k+1)\cdot v(k+1)\\ &= \hat{p}(k+1|k) + K(k+1)\cdot [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))]\\ &= \hat{p}(k+1|k) + K(k+1)\cdot [z_j(k+1) - z_t] \end{aligned} \quad (5.57)$$

corresponding to equation (5.44).

5.6.3.3 Case study: Kalman filter localization with line feature extraction

The Pygmalion robot at EPFL is a differential-drive robot that uses a laser rangefinder as its primary sensor [37, 38]. In contrast to both Dervish and Rhino, the environmental representation of Pygmalion is continuous and abstract: the map consists of a set of infinite lines describing the environment. Pygmalion's belief state is, of course, represented as a Gaussian distribution since this robot uses the Kalman filter localization algorithm. The value of its mean position μ is represented to a high level of precision, enabling Pygmalion to localize with very high precision when desired. Below, we present details for Pygmalion's implementation of the five Kalman filter localization steps. For simplicity we assume that the sensor frame $\{S\}$ is equal to the robot frame $\{R\}$. If not specified all the vectors are represented in the world coordinate system $\{W\}$.

1. Robot position prediction. At the time increment k the robot is at position $p(k) = \begin{bmatrix} x(k) & y(k) & \theta(k) \end{bmatrix}^T$ and its best position estimate is $\hat{p}(k|k)$. The control input $u(k)$ drives the robot to the position $p(k+1)$ (figure 5.29).

The robot position prediction $\hat{p}(k+1)$ at the time increment $k+1$ can be computed from the previous estimate $\hat{p}(k|k)$ and the odometric integration of the movement. For the differential drive that Pygmalion has we can use the model (odometry) developed in section 5.2.4:

$$\hat{p}(k+1|k) = \hat{p}(k|k) + u(k) = \hat{p}(k|k) + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2}\cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2}\sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \quad (5.58)$$

with the updated covariance matrix

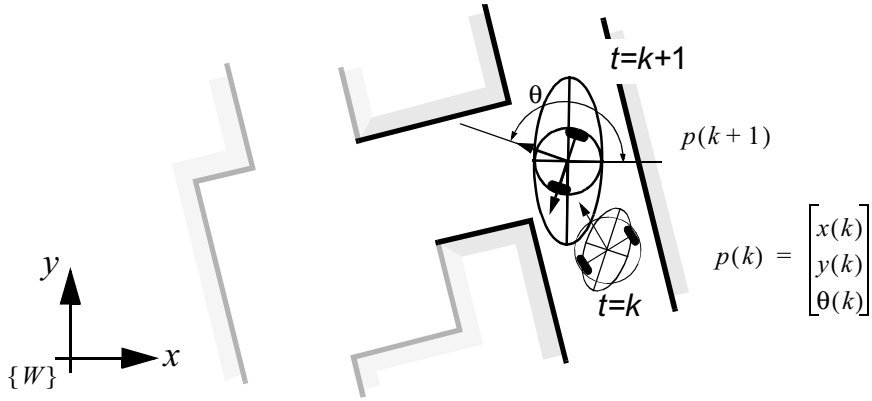

Figure 5.29
Prediction of the robot's position (thick) based on its former position (thin) and the executed movement. The ellipses drawn around the robot positions represent the uncertainties in the *x,y* direction (e.g.; 3σ). The uncertainty of the orientation θ is not represented in the picture.

$$\Sigma_p(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T \tag{5.59}$$

where

$$\Sigma_u = cov(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix} \tag{5.60}$$

2. Observation. For line-based localization, each single observation (i.e., a line feature) is extracted from the raw laser rangefinder data and consists of the two line parameters $\beta_{0,j}$, $\beta_{1,j}$ or α_j, r_j (figure 4.36) respectively. For a rotating laser rangefinder, a representation in the polar coordinate frame is more appropriate and so we use this coordinate frame here:

$$z_j(k+1) = {}^R\begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} \tag{5.61}$$

After acquiring the raw data at time *k*+1, lines and their uncertainties are extracted (figure 5.30a, b). This leads to n_0 observed lines with $2n_0$ line parameters (figure 5.30c) and a covariance matrix for each line that can be calculated from the uncertainties of all the

measurement points contributing to each line as developed for line extraction in section 4.3.1.1:

$$\Sigma_{R,j} = \begin{bmatrix} \sigma_{\alpha\alpha} & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_{rr} \end{bmatrix}_j \tag{5.62}$$

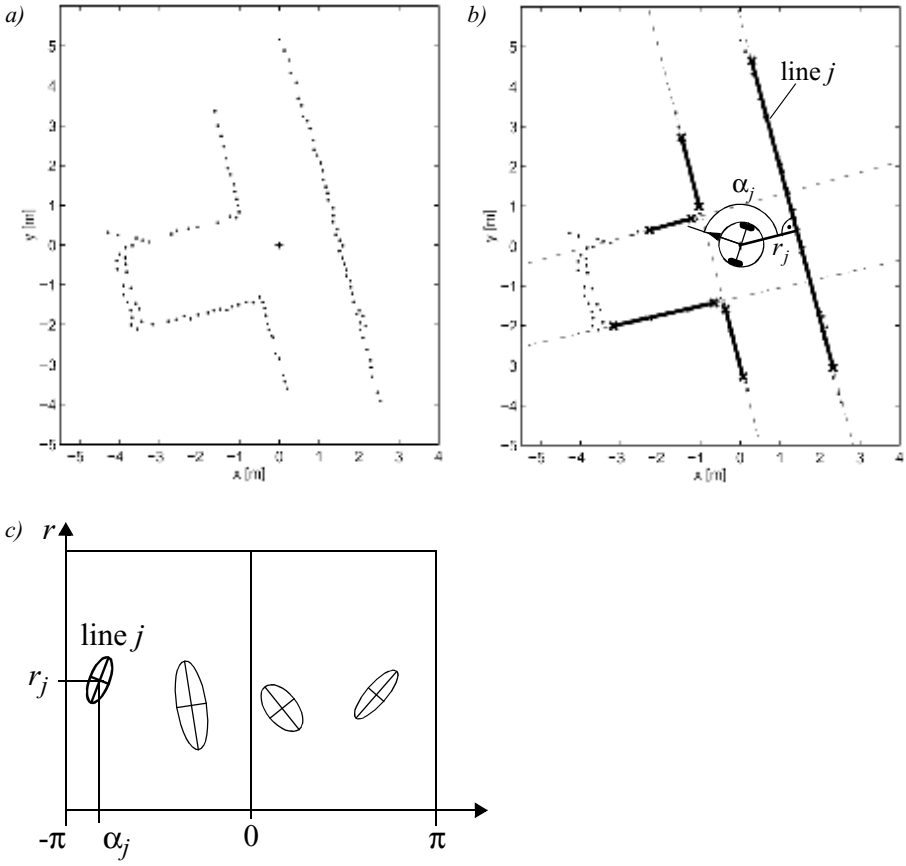

Figure 5.30
Observation: From the raw data (a) acquired by the laser scanner at time $k + 1$, lines are extracted (b). The line parameters α_j and r_j and its uncertainties can be represented in the model space (c).

3. Measurement prediction. Based on the stored map and the predicted robot position $\hat{p}(k|k)$, the measurement predictions of expected features $z_{t,i}$ are generated (figure 5.31). To reduce the required calculation power, there is often an additional step that first selects the possible features, in this case lines, from the whole set of features in the map. These lines are stored in the map and specified in the world coordinate system $\{W\}$. Therefore they need to be transformed to the robot frame $\{R\}$:

$$ {}^{W}z_{t,i} = {}^{W}\begin{bmatrix} \alpha_{t,i} \\ r_{t,i} \end{bmatrix} \rightarrow {}^{R}z_{t,i} = {}^{R}\begin{bmatrix} \alpha_{t,i} \\ r_{t,i} \end{bmatrix} \tag{5.63} $$

According to figure (5.31), the transformation is given by

$$ \begin{aligned} \hat{z}_i(k+1) = {}^{R}\begin{bmatrix} \alpha_{t,i} \\ r_{t,i} \end{bmatrix} &= h_i(z_{t,i}, \hat{p}(k+1|k)) \\ &= \begin{bmatrix} {}^{W}\alpha_{t,i} - {}^{W}\hat{\theta}(k+1|k) \\ {}^{W}r_{t,i} - ({}^{W}\hat{x}(k+1|k)\cos({}^{W}\alpha_{t,i}) + {}^{W}\hat{y}(k+1|k)\sin({}^{W}\alpha_{t,i})) \end{bmatrix} \end{aligned} \tag{5.64} $$

and its Jacobian ∇h_i by

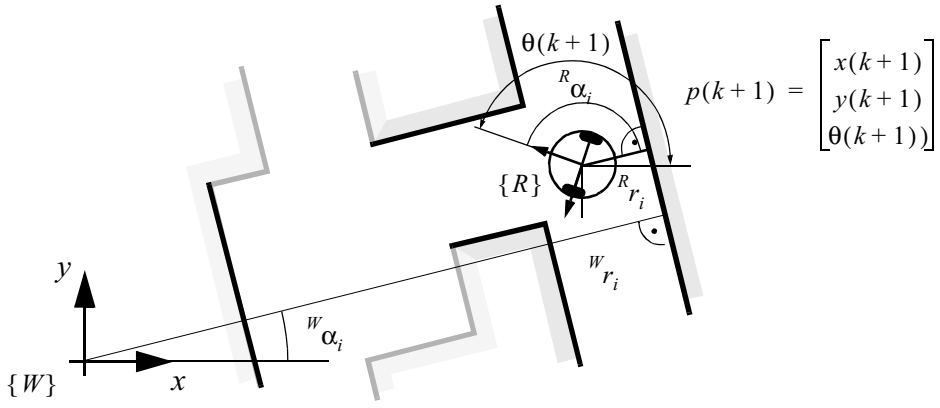

Figure 5.31
Representation of the target position in the world coordinate frame $\{W\}$ and robot coordinate frame $\{R\}$.

$$\nabla h_i = \begin{bmatrix} \frac{\partial \alpha_{t,i}}{\partial \hat{x}} & \frac{\partial \alpha_{t,i}}{\partial \hat{y}} & \frac{\partial \alpha_{t,i}}{\partial \hat{\theta}} \\ \frac{\partial r_{t,i}}{\partial \hat{x}} & \frac{\partial r_{t,i}}{\partial \hat{y}} & \frac{\partial r_{t,i}}{\partial \hat{\theta}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ -\cos {}^{W}\alpha_{t,i} & -\sin {}^{W}\alpha_{t,i} & 0 \end{bmatrix} \tag{5.65}$$

The measurement prediction results in predicted lines represented in the robot coordinate frame (figure 5.32). They are uncertain, because the prediction of robot position is uncertain.

4. Matching. For matching, we must find correspondence (or a pairing) between predicted and observed features (figure 5.33). In our case we take the Mahalanobis distance

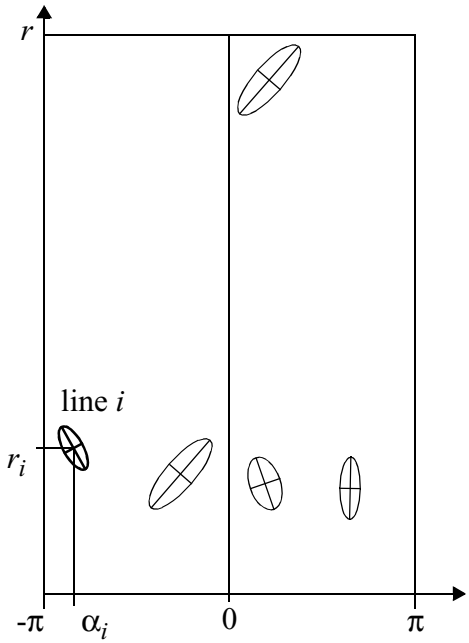

Figure 5.32
Measurement predictions: Based on the map and the estimated robot position the targets (visible lines) are predicted. They are represented in the model space similar to the observations.

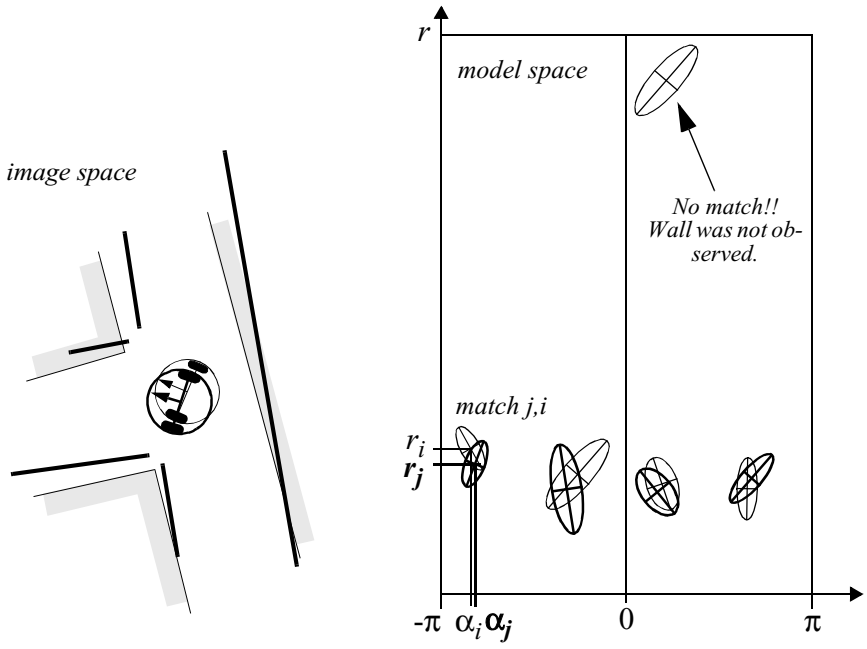

Figure 5.33
Matching: The observations (thick) and measurement prediction (thin) are matched and the innovation and its uncertainties are calculated.

$$v_{ij}^T(k+1) \cdot \Sigma_{IN,\,ij}^{-1}(k+1) \cdot v_{ij}(k+1) \leq g^2 \tag{5.66}$$

with

$$\begin{aligned} v_{ij}(k+1) &= [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))] \\ &= \begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} - \begin{bmatrix} {}^W\alpha_{t,\,i} - {}^W\hat{\theta}(k+1|k) \\ {}^W r_{t,\,i} - ({}^W\hat{x}(k+1|k)\cos({}^W\alpha_{t,\,i}) + {}^W\hat{y}(k+1|k)\sin({}^W\alpha_{t,\,i})) \end{bmatrix} \end{aligned} \tag{5.67}$$

$$\Sigma_{IN,\,ij}(k+1) = \nabla h_i \cdot \Sigma_p(k+1|k) \cdot \nabla h_i^T + \Sigma_{R,\,i}(k+1) \tag{5.68}$$

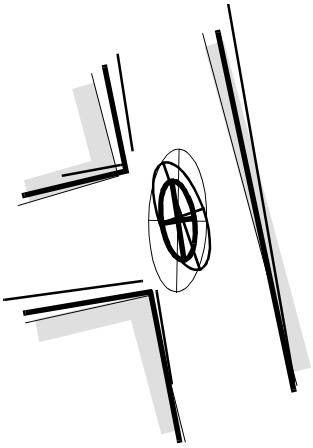

Figure 5.34
Kalman filter estimation of the new robot position: By fusing the prediction of robot position (thin) with the innovation gained by the measurements (thick) we get the updated estimate $\hat{p}(k|k)$ of the robot position (very thick).

to enable finding the best matches while eliminating all other remaining observed and predicted unmatched features.

5. Estimation. Applying the Kalman filter results in a final pose estimate corresponding to the weighted sum of (figure 5.34)

- the pose estimates of each matched pairing of observed and predicted features;
- the robot position estimation based on odometry and observation positions.

5.7 Other Examples of Localization Systems

Markov localization and Kalman filter localization have been two extremely popular strategies for research mobile robot systems navigating indoor environments. They have strong formal bases and therefore well-defined behavior. But there are a large number of other localization techniques that have been used with varying degrees of success on commercial and research mobile robot platforms. We will not explore the space of all localization systems in detail. Refer to surveys such as [5] for such information.

There are, however, several categories of localization techniques that deserve mention. Not surprisingly, many implementations of these techniques in commercial robotics