AA 274: Principles of Robotic Autonomy — Winter 2018

Lecture 12: Localization III: Markov localization and EKF-localization

Scribes: Eric Chang, Diana Chin, Laura Matloff, Will Roderick, Navjot Singh, Jiahui Wang

12.1 Introduction

The aim of these notes is to help you learn about Markov localization with an emphasis on Extended Kalman Filter (EKF) localization. These probabilistic map-based localization techniques address realistic issues of noise and uncertainty in robotic data and often out-perform alternative techniques when applied to the real world.

12.2 Casting the Localization Problem within a Bayesian Filtering Framework

12.2.1 Terminology

As in the general filtering context, at time t, the state is given by x_t, the control input is given by u_t, and the measurements are given by the observation z_t. For a differential drive robot equipped with a laser range-finder (returning a set of range r_i and bearing ϕ_i measurements), we have

$$x_t = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \qquad u_t = \begin{pmatrix} v \\ \omega \end{pmatrix} \qquad z_t = \left\{ \begin{pmatrix} r_i \\ \phi_i \end{pmatrix} \right\}_i$$

12.2.2 Strategy and Ingredients of probabilistic map-based localization

The general strategy for the robot localization problem occurs in two steps: First, during the *prediction* (or *action*) *update*, the robot estimates its position through proprioception (such as encoders or dead reckoning) The uncertainty of the robot during this phase increases due to the accumulation of odometric error (through integrating over time). The second step is the *perception* (or *measurement*, or *correction*) *update*, where the robot uses exteroceptive sensors (such as ultrasonic, laser, camera) to correct its earlier estimated prediction. The uncertainty of the robot configuration shrinks. By combining these two steps, a robot can localize with respect to its map.

In order to solve the robot localization problem, the following ingredients are needed: [1]

1. **Initial probability distribution**, *bel(x_0)* for the initial robot location.
2. **Map of the environment**. If not known *a priori*, then it can be built.
3. **Data** from robotic sensors including the observation z_t and the control input u_t.

4. **Probabilistic motion model**. Derived from robotic kinematics, the current location, x, is a function of previous location, x_t, and control input u_t, with additional modeled error distributions.
5. **Probabilistic measurement model**. This model is derived from the sensor measurements of the robot. It consists of the measurement function h depends on the environment map and the robot location and an added noise term such that the probability distribution peaks at the noise-free value.

12.2.3 Maps

A key ingredient to instantiate Bayes filtering in the context of localization is a map. A map m is a list of objects in the environment along with their properties, given by

$$m = \{m_1, m_2, \ldots, m_N\}$$

Each m_i is an object that encapsulates some property of the environment. The two types of maps we will focus on here are location-based maps and feature-based maps. As will be discussed further in the next sections, different map types typically have a trade-off in computational efficiency and explicitness.

12.2.3.1 Location-based maps

For location-based maps, an index i corresponds to a specific location (and hence, they are volumetric). Figure 12.1 shows two examples of location-based maps.

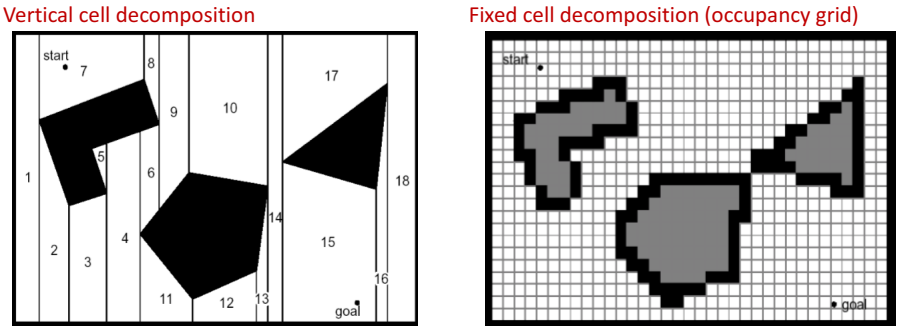


Figure 12.1: Two examples of location-based maps.

The first example of a location-based map employs vertical cell decomposition. This method essentially sweeps a vertical line through the width of the environment, and every time a corner is encountered, a cell is built. The result is a set of trapezoidal cells as shown in Fig. 12.1, left. Each element of the map vector would be one of the cells. It is important to note that this technique can only describe the location of the robot using the presence or absence of the robot in a cell, but cannot give information on where within a cell the robot may be.

The second common example of a location-based map uses fixed cell decomposition (Fig. 12.1, right). In this grid map, the ith element of the map corresponds to the vector $\boldsymbol{x}_i$ position of the ith cell of the grid map. Thus, each cell represents a map vector.

As compared the the fixed cell decomposition, the vertical cell decomposition is more efficient, as it abstracts the environment as a graph with nodes and edges. However, fixed cell decomposition can have better spatial resolution at the expense of additional computational complexity. Both techniques can give both the presence and absence of an object, which cannot be done with feature-based maps.

12.2.3.2 Feature-based maps

For feature-based maps, an index i is a feature index, and m_i contains, next to the properties of a feature, the Cartesian location of that feature. One can think of this type of map as a collection of landmarks, such as points or lines. Figure 12.2 gives two examples of feature-based maps.

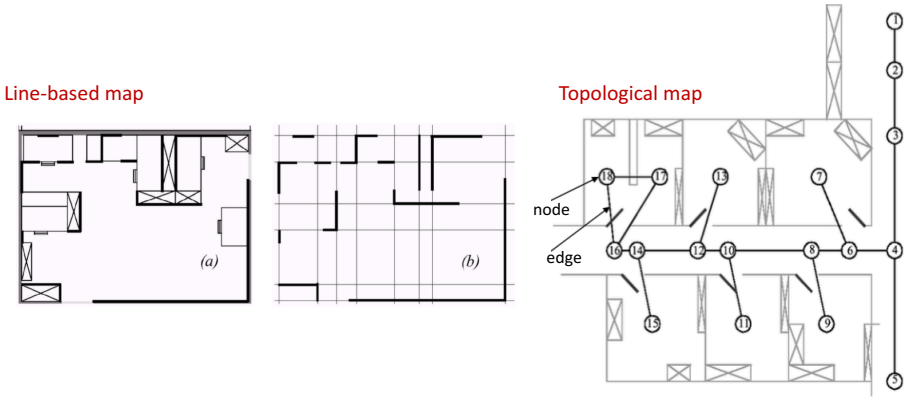


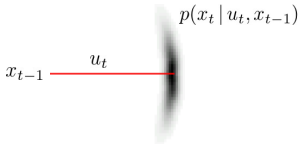
Figure 12.2: Two examples of feature-based maps.

The first example, the lined-based map, uses a line representation of the environment and is commonly used in structured environments like buildings. This is the type of map that we will focus on in this class, but the concepts covered on filtering and localization apply to all maps.

The second example is the topological map, which, rather than directly measuring geometric environmental quantities, abstracts the environment as a collection of high level features. These features could be the position of a chair or a lamp, for example.

These types of abstractions can improve the computational efficiency but feature-based maps cannot give information on the presence and absence of a specific object in the environment.

12.2.4 State Transition



As we have seen previously, the motion model is probabilistic: $p(x_t|u_t, x_{t-1})$ describes the posterior distribution over the kinematic states that the robot assumes when executing control u_t when starting from x_{t-1}. An illustration of this concept is shown in Figure 12.2.4. However, unlike what we have seen previously, we introduce the map, which plays a role in the forward propagation of the dynamics. It is important to note that $p(x_t|u_t, x_{t-1}) \neq p(x_t|u_t, x_{t-1}, m)$. To understand why, consider the possible future states of the robot when it is next to a wall. If we do not consider the map, the model may erroneously give a non zero probability that the robot can travel through the obstacle.

To avoid sampling from the state transition function, which is hard to do, a common technique is to make the approximation:

$$p(x_t|u_t, x_{t-1}, m) \approx \eta \frac{p(x_t|u_t, x_{t-1})p(x_t|m)}{p(x_t)}$$

The derivation of this approximation is as follows. Using Bayes theorem,

$$p(x_t|x_{t-1}, u_t, m) = \frac{p(m|x_t, x_{t-1}, u_t)p(x_t|x_{t-1}, u_t)}{p(m|x_{t-1}, u_t)}$$

Because $p(m|x_{t-1}, u_t)$ is independent of x_t we can include it in a normalizer, η', so

$$p(x_t|x_{t-1}, u_t, m) = \eta' p(m|x_t, x_{t-1}, u_t)p(x_t|x_{t-1}, u_t)$$

Now we make the approximation that $p(m|x_t, x_{t-1}, u_t) \approx p(m|x_t)$, which neglects where we are coming from (discarding the information that says along a given motion, we might have an obstacle).

$$p(x_t|x_{t-1}, u_t, m) = \eta' p(m|x_t)p(x_t|x_{t-1}, u_t)$$

The last step involves using Bayes rule again to write $p(m|x_t) = \frac{p(x_t|m)p(m)}{p(x_t)}$. Because $p(m)$ is independent of x_t, we can include it in a normalizer: $\eta = \eta' p(m)$.

Finally,

$$p(x_t|u_t, x_{t-1}, m) \approx \eta \frac{p(x_t|u_t, x_{t-1})p(x_t|m)}{p(x_t)} \tag{12.1}$$

η, the normalizing factor, is equal to one divided by the sum of the total probability of the rest of expression so that the resulting probability density function sums to 1.

In general, the prior with respect to x_t is uniform so the denominator in the final expression also gets included as a constant in the normalizer. Then, the terms that remain are the two probability functions in the numerator. The first is the probability distribution of the future state of the robot as if there were no obstacles. The second is the probability of x given m, which is essentially a consistency check that gives the plausibility of a state x_t given a map. For example, in a grid world representation, the result could be a binary random variable, where the probability that the robot is in the same cell as an obstacle is 0 and otherwise the result is 1. This approximation essentially computes the state transition assuming no obstacles and then applies a consistency check.

Next, we ground the measurement model in the map. The measurement model is probabilistic $p(z_t|x_t, m)$ and we have to condition with respect to a map because the observation also depends on the map. Sensors usually generate more than one measurement when queried, so we have batches of measurements $z_t = \{z_t^1, \ldots, z_t^K\}$. This is the main difference with respect to the Bayes filter (where we were considering sequential measurements). Typically, for simplicity, we assume that these measurements are independent so that

$$p(z_t|x_t, m) = \prod_{k=1}^{K} p(z_t^k|x_t, m)$$

12.3 Markov Localization

Markov localization uses a specified probability distribution across all possible robot positions. This high-level instantiation of the Bayes filter in the context of localization is fairly straightforward as follows.

Data: $bel(x_{t-1}), u_t, z_t, m$
Result: $bel(x_t)$
foreach x_t **do**
  $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx_{t-1}$;
  $bel(x_t) = \eta p(z_t \mid x_t, m)\, \overline{bel}(x_t)$;
end
Return $bel(x_t)$

We input our belief pose at $t-1$ given by $bel(x_{t-1})$, our control u_t, our measurement z_t, and the conditioning over the map m. The Markov localization model is conceptually identical to the Bayes filter except for the inclusion of m.

Markov localization has the goal of iteratively propagating forward the belief over the pose of the robot in two steps. The first step is a prediction step that leverages the knowledge of the state transition function of the robot to go from $bel(x_{t-1})$ to $\overline{bel}(x_t)$. The second step is the correction step that accounts for the measurement feedback. Figure 12.3 illustrates an example of this algorithm.

As is the case for Bayes filters, we must initialize the Markov localization algorithm with an initial belief $bel(x_0)$. For position tracking, if the initial pose is known,

$$bel(x_0) = \begin{cases} 1 & \text{if } x_0 = \overline{x}_0 \\ 0 & \text{otherwise} \end{cases}$$

If the initial pose is partially known,

$$bel(x_0) \sim \mathcal{N}(\overline{x}_0, \Sigma_0)$$

.

For global localization, where the initial pose is unknown, the belief is initialized as a uniform distribution

$$bel(x_0) = 1/|X|$$

To make the Markov localization algorithm tractable, we need to add some structure to the representation of $bel(x_t)$. We can consider three representations. The next section will cover a Gaussian representation in the context of EKF localization. Grid representations and particle filter representations will be covered in the next lecture.

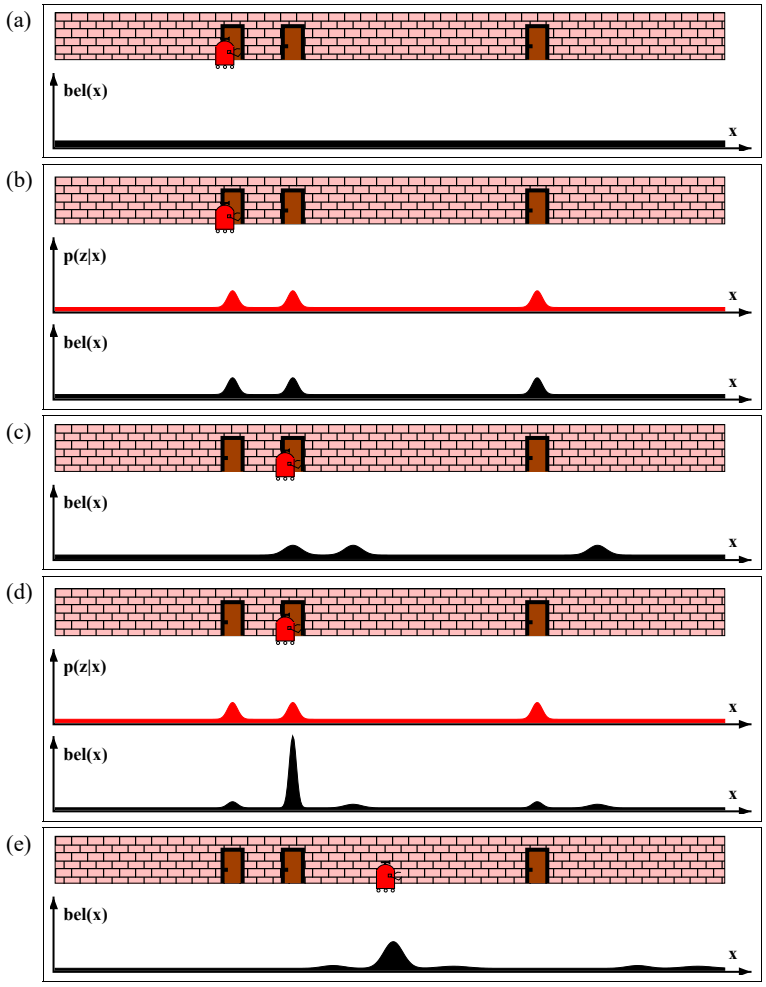


Figure 12.3: Markov localization algorithm illustration. We consider an example with a robot moving along a wall with three doors (a) The belief is initialized with a uniform distribution. (b) The robot moves to the first door and makes an observation. We apply the correction step to update our belief (c) The robot moves to the second door. The shifted belief is flattened because there is uncertainty with how much we moved. (d) We take a second observation, apply the correction, and obtain a large peak in the posterior belief. (e) The robot's belief after moving further down the wall

12.4 Extended Kalman Filter (EKF) Localization

The key idea of EKF localization is to represent the belief $bel(x_t)$ by its first and second moment, μ_t and Σ_t. We will assume that a feature-based map is available, consisting of point landmarks, given by

$$m = \{m_1, m_2, \dots\}, \qquad m_j = (m_{j,x}, m_{j,y})$$

where each m_j encapsulates the location of the landmark in the global coordinate frame. We also assume that we have a sensor that can measure the range r and the bearing ϕ of the landmarks relative to the robot's local coordinate frame.

The range and bearing sensors measure a set of independent features at time t

$$z_t = \{z_t^1, z_t^2, \dots\} = \{(r_t^1, \phi_t^1), (r_t^2, \phi_t^2), \dots\}$$

where each measurement z_t^i contains the range r_t^i and bearing ϕ_t^i. We also instantiate a motion model assuming a differential drive robot with states (x, y, θ).

Now that we've made assumptions about the map, measurements, and the robot motion, we can instantiate the measurement model which will allow us to re-derive the equations for the Markov localization filter. The measurement model tells us the likelihood of a given measurement given our state. Therefore, assuming that the i-th measurement at time t corresponds to the j-th landmark in the map m, the measurement model is

$$\begin{pmatrix} r_t^i \\ \phi_t^i \end{pmatrix} = h(x_t, j, m) + \mathcal{N}(0, Q_t)$$

where $\mathcal{N}(0, Q_t)$ models Gaussian noise and

$$h(x_t, j, m) = \begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \end{pmatrix} \tag{12.2}$$

12.4.1 Data Association

The data association problem is the potential uncertainty about the identity of a landmark. For instance, we may be given a range and bearing, but these measurements are not very meaningful if we do now know what landmark they are given with respect to. As a result, for a map with N total landmarks, we define a *correspondence variable* $c_t^i \in 1, ..., N+1$ that relates measurement z_t^i and landmark m_j, where

- $c_t^i = j \leq N$ if the i-th measurement at time t corresponds to the j-th landmark
- $c_t^i = N + 1$ if a measurement does not correspond to any landmark

There are two versions of the localization problem - when the correspondence variables are known, and when they are not known. We generally only deal with the second case in practice, but to gain insight, we start by discussing the first version.

12.4.2 EKF Localization with Known Correspondences

The EKF localization algorithm is derived from the EKF filter, but with two main differences - measurements are now associated with landmarks, and multiple measurements are processed at the same time. We begin again with our differential drive robot and assumption of the stochastic motion model (with some process disturbance ϵ that has a Gaussian distribution) and Jacobian G:

$$x_t = g(u_t, x_{t-1}) + \epsilon_t, \qquad \epsilon \sim \mathcal{N}(0, R_t), \qquad G_t := J_g(\mu_t, \mu_{t-1})$$

The (nonlinear) range and bearing measurement model $h(x_i, j, m)$ now includes the index j of the landmark with respect to which we are taking the measurements:

$$z_t^i = h(x_i, j, m) + \delta_t, \qquad \delta_t \sim \mathcal{N}(0, Q_t), \qquad H_t^i := \frac{\partial h(\bar{\mu}_t, j, m)}{\partial x_t}$$

Here δ is again the measurement noise, and H is the measurement Jacobian. From differentiating 12.2,

$$\frac{\partial h(\bar{\mu}_t, j, m)}{\partial x_t} = \begin{pmatrix} \frac{\partial r_t^i}{\partial \bar{\mu}_{t,x}} & \frac{\partial r_t^i}{\partial \bar{\mu}_{t,y}} & \frac{\partial r_t^i}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,x}} & \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,y}} & \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,\theta}} \end{pmatrix} = \begin{pmatrix} -\frac{m_{j,x}-\bar{\mu_t}_{,x}}{\sqrt{(m_{j,x}-\bar{\mu_t}_{,x})^2+(m_{j,y}-\bar{\mu_t}_{,y})^2}} & -\frac{m_{j,y}-\bar{\mu_t}_{,y}}{\sqrt{(m_{j,x}-\bar{\mu_t}_{,x})^2+(m_{j,y}-\bar{\mu_t}_{,y})^2}} & 0 \\ \frac{m_{j,y}-\bar{\mu_t}_{,y}}{(m_{j,x}-\bar{\mu_t}_{,x})^2+(m_{j,y}-\bar{\mu_t}_{,y})^2} & -\frac{m_{j,x}-\bar{\mu_t}_{,x}}{\sqrt{(m_{j,x}-\bar{\mu_t}_{,x})^2+(m_{j,y}-\bar{\mu_t}_{,y})^2}} & -1 \end{pmatrix}$$

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{pmatrix}$$

The EKF localization algorithm is shown below. The key differences from the Bayes/EKF filter are the new inputs (the correspondence variables c_t and map m), and the fact that we are now considering batches of measurements. As with the Bayes filter, we start with the prediction step, in which the mean and variance are propagated using Taylor series expansions. We then carry out the correction step by exploiting the conditional independence assumption for our batch of measurements,

$$p(z_t|x_t, c_t, m) = \prod_i p(z_t^i|x_t, c_t^i, m)$$

which allows us to incrementally add information (i.e. process the measurements sequentially) as if there was no motion between measurements.

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t, c_t, m$
Result: (μ_t, Σ_t)
$\overline{\mu}_t = g(u_t, \mu_{t-1})$;
$\overline{\Sigma}_t = G_t \Sigma_{t-1} {G_t}^T + R_t$; Uncertainty about where we are at time t-1 before taking measurements into account = uncertainty about pose at time t-1 + uncertainty about motion model
foreach $z_t^i = (r_t^i, \phi_t^i)^T$ **do**
  $j = c_t^i$; Identify landmark that corresponds with each measurement
  $\hat{z}_t^i = \begin{pmatrix} \sqrt{(m_{j,x} - \overline{\mu}_{t,x})^2 + (m_{j,y} - \overline{\mu}_{t,y})^2} \\ \text{atan2}(m_{j,y} - \overline{\mu}_{t,y}, m_{j,x} - \overline{\mu}_{t,x}) - \overline{\mu}_{t,\theta} \end{pmatrix}$; Predict measurement by applying the measurement model on the expectation of the predicted belief
  $S_t^i = H_t^i \overline{\Sigma}_t [H_t^i]^T + Q_t$; Measurement covariance = uncertainty about robot pose + uncertainty about measurement process
  $K_t^i = \overline{\Sigma}_t [H_t^i]^T [S_t^i]^{-1}$;
  $\overline{\mu}_t = \overline{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$; Use Kalman gain to correct the predicted belief with the innovation vector (difference between actual and predicted measurement)
  $\overline{\Sigma}_t = (I - K_t^i H_t^i) \overline{\Sigma}_t$;
end
$\mu_t = \overline{\mu}_t$ and $\Sigma_t = \overline{\Sigma}_t$;
Return (μ_t, Σ_t)

The EKF-localization steps are illustrated for a differential drive robot in the following figures:

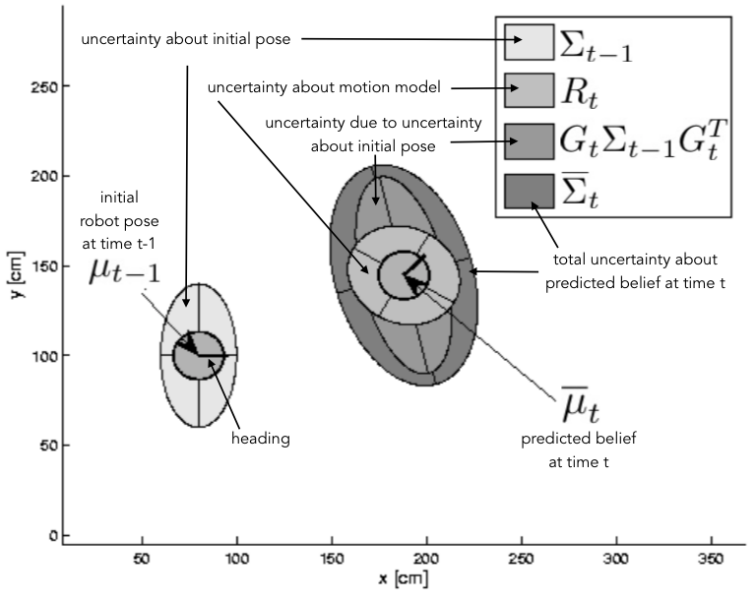


Figure 12.4: The prediction step. Observations measure the relative distance (range) and heading (bearing) of the robot to a landmark. We assume here that the robot detects only one landmark at a time.

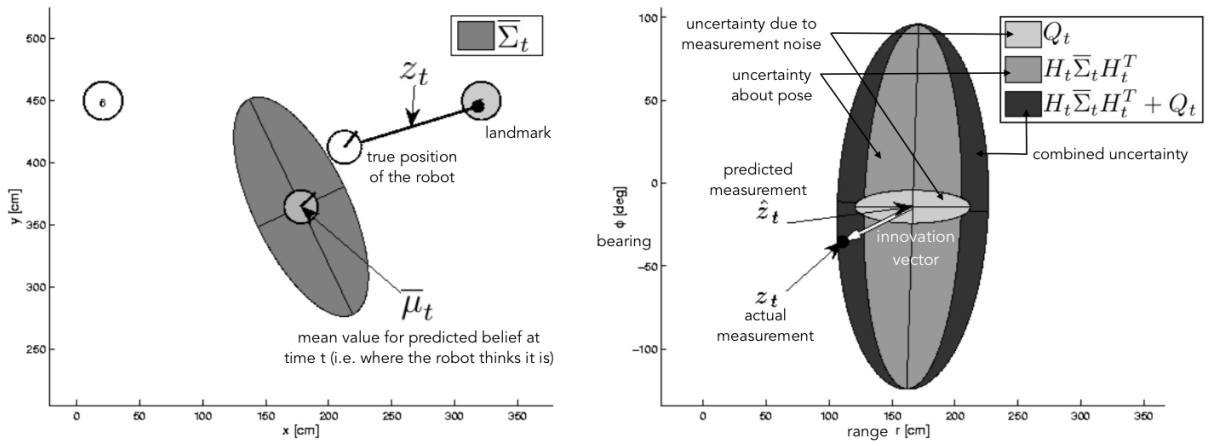


Figure 12.5: The measurement prediction step. In this example, the measured range is much shorter than what was expected based on the predicted belief. The innovation vector, represented by the white arrow in the diagram on the right, shows the difference between the actual and predicted measurements.

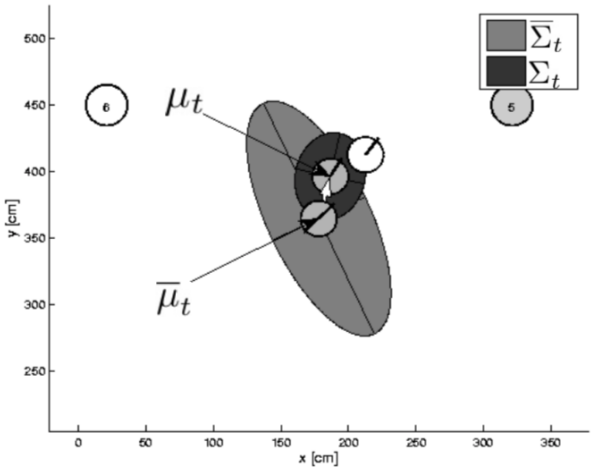


Figure 12.6: The correction step. The expectation $\bar{\mu}_t$ is corrected by moving in the direction indicated by the white innovation vector (also shown in Fig. 12.5), resulting in an updated position μ_t that is closer to the true position (indicated by the white circle).

12.4.3 EKF Localization with Unknown Correspondences

For EKF localization with unknown correspondences, we must jointly estimate the correspondence variables and use these estimates to implement the Markov filter. The simplest way to determine the identity of a landmark during localization is to use maximum likelihood estimation, in which the most likely value of the correspondences c_t is determined by maximizing the data likelihood:

$$\hat{c}_t = \arg\max_{c_t} p(z_t|c_{1:t}, m, z_{1:t-1}, u_{1:t})$$

In other words, we pick the set of correspondence variables that maximizes the probability of getting the measurement that we see, given the history of correspondence variables, the map, the history of measurements, and the history of controls. The value of c_t is then taken for granted, so this method can be quite brittle; any mistake made during this process will propagate into the future.

The challenge with this method is that each element can take on many values (equal to the number of landmarks), and the number of possible combinations is exponential. The optimization problem is therefore over an exponentially large physical space for a nonlinear function. As a result, we must resort to applying reasonable approximations with a model of independent measurements, and perform maximization separately for each z_t^i.

12.4.4 Estimating the Correspondence Variable

The first step to solving the set of correspondence variables is to optimize the correspondence variables one by one:

$$p(z_t^i|c_{1:t}, m, z_{1:t-1}, u_{1:t})$$

Essentially, we want to find the likelihood of z_t^i given the conditions at c, m, z, and u. In order to do so, we use the total probability theorem. Through this method, we are optimizing the likelihood

of each individual event instead of a joint optimization of all likelihood events at once. After algebraic manipulation, this translates to performing the following calculation:

$$p(z_t^i|c_{1:t}, m, z_{1:t-1}, u_{1:t}) \approx N(h(\bar{\mu}_t, c_t^i, m), H_t^i \bar{\Sigma}_t [H_t^i]^T + Q_t)$$

What we get is that the probability of z_t^i is approximately equal to the Gaussian distribution with mean equal to the predicted measurement. The next step is to find the correspondences using this Gaussian distribution with the following estimation:

$$\hat{c}_t^i = argmax p(z_t^i|c_{1:t}, m, z_{1:t-1}, u_{1:t}) \approx argmax N(z_t^i; h(\bar{\mu}_t, c_t^i, m), H_t \bar{\Sigma}_t H_t^T + Q_t)$$

Very similar to the EKF with known correspondences, we apply an algorithm very similar to the one for EKF localization for known parameters. This includes a maximum likelihood estimator for the correspondence variables, and we are now no longer assuming the correspondence variables are inputs but rather optimizing for their values.The general outline of the algorithm is to do prediction, estimation, correspondences between measurements and landmarks, and finally correction. Examples of some common landmarks are lines, corners, and distinct patterns.

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t, m$
Result: (μ_t, Σ_t)
$\overline{\mu}_t = g(u_t, \mu_{t-1})$;
$\overline{\Sigma}_t = G_t \Sigma_{t-1} {G_t}^T + R_t$;
foreach $z_t^i = (r_t^i, \phi_t^i)^T$ **do**
  foreach *landmark k in the map* **do**
    $\hat{z}_t^k = \begin{pmatrix} \sqrt{(m_{k,x} - \overline{\mu}_{t,x})^2 + (m_{k,y} - \overline{\mu}_{t,y})^2} \\ \text{atan2}(m_{k,y} - \overline{\mu}_{t,y}, m_{k,x} - \overline{\mu}_{t,x}) - \overline{\mu}_{t,\theta} \end{pmatrix}$;
    $S_t^k = H_t^k \overline{\Sigma}_t [H_t^k]^T + Q_t$;
  end
  $j(i) = \arg\max_k \mathcal{N}(z_t^i; \hat{z}_t^k, S_t^k)$
  $K_t^i = \overline{\Sigma}_t [H_t^{j(i)}]^T [S_t^{j(i)}]^{-1}$;
  $\overline{\mu}_t = \overline{\mu}_t + K_t^i (z_t^i - \hat{z}_t^{j(i)})$;
  $\overline{\Sigma}_t = (I - K_t^i H_t^{j(i)}) \overline{\Sigma}_t$;
end
$\mu_t = \overline{\mu}_t$ and $\Sigma_t = \overline{\Sigma}_t$;
Return (μ_t, Σ_t)

Correspondence estimation

12.5 A Taxonomy of Localization problems

This is a summary of localization problems from Ref. [2] Chapter 7.2 to help us understand the difficulty of a localization problem.

Local vs. Global localization

Local versus global localization can be characterized by the type of knowledge that is available initially and at run-time.

- **Position tracking** - the *initial* robot pose is *known*. The pose error is small. The uncertainty of Position tracking problem is local problem and confined to region near the robot's true pose, usually can be approximated by a Gaussian distribution.
- **Global localization** - the *initial* robot pose is *unknown*. Unimodal probability distributions (e.g., a Gaussian) are usually inappropriate.Thus global localization is more difficult than position tracking.
- **Kidnapped robot problem** During operation, the robot can get kidnapped and teleported to some other unknown location, thus making localization more difficult. In this case, the robot might believe it knows where it is, while in reality, it does not. Testing a localization algorithm by kidnapping the robot measures its ability to recover from global localization failures.

Static vs. Dynamic Environments

- **Static environments** The only variable quantity (state) is the robots pose. All objects except the robot remain at the same location forever.
- **Dynamic environments** The states of other objects whose location or configuration change over time. Harder than static environments.

Passive vs. Active Approaches

Passive versus active approaches distinguishes whether or not the localization algorithm controls the motion of the robot.

- **Passive localization** The robot control is independent of the localization process.
- **Active localization** Active localization algorithms control the robot to minimize the localization error and/or the costs arising from moving a poorly localized robot into a hazardous place.

Active localization algorithm typically perform better than passive ones. But active localization needs to be combined with localization goals and a controlled robot or be built on top of a passive localization algorithm.

Single-Robot vs. Multi-Robot

The difference is the number of robots.

- **Single-robot localization** All data is collected at a single robot platform and there is no communication issue.
- **Multi-robot localization** If all robots work independently, then single-robot localization can be applied for muti-robot localization. The situation becomes complicated when communication happens between them.

References

[1] I. R. Nourbakhsh R. Siegwart and D. Scaramuzza. *Introduction to Autonomous Mobile Robots, Chapters 5.6.4 5.6.8, and 5.7*. 2nd ed. MIT press, 2011.

[2] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics, Chapter 7.2-7.6*. MIT press, 2005.