ECEN426
Advanced Mechatronic Systems
Simultaneous Localisation and Mapping – SLAM

Christopher Hollitt

October 14, 2021

Part I

Localisation

Robot localisation

Consider the model for the robot. That is, we wish to predict the next state of the robot given where we currently are, and what we intend to do with the command signal.

$$\begin{aligned} s_{\mathrm{x}}(t+1) &= s_{\mathrm{x}}(t) + u_1(t)\cos\theta(t) + w_1(t)\cos\theta(t) \\ s_{\mathrm{y}}(t+1) &= s_{\mathrm{y}}(t) + u_1(t)\sin\theta(t) + w_1(t)\sin\theta(t) \\ \theta(t+1) &= \theta(t) + u_2(t) + w_2(t) \end{aligned}$$

With w_1 and w_2 the noise acting in the forward, and angular speed directions.

Let's rewrite these equations in terms of the state variables defined by $\left[x_1, x_2, x_3\right]^{\mathsf{T}} := \left[s_{\mathrm{x}}, s_{\mathrm{y}}, \theta\right]^{\mathsf{T}}$. Furthermore, let's denote the position of the i-th beacon with $\left[z_1^{(i)}, z_2^{(i)}\right]^{\mathsf{T}}$.

$$\begin{aligned} x_1(t+1) &= f_1(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) = x_1(t) + u_1\cos x_3 + w_1\cos x_3 \\ x_2(t+1) &= f_2(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) = x_2(t) + u_1\sin x_3 + w_1\sin x_3 \\ x_3(t+1) &= f_3(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) = x_3(t) + u_2 + w_2 \end{aligned}$$

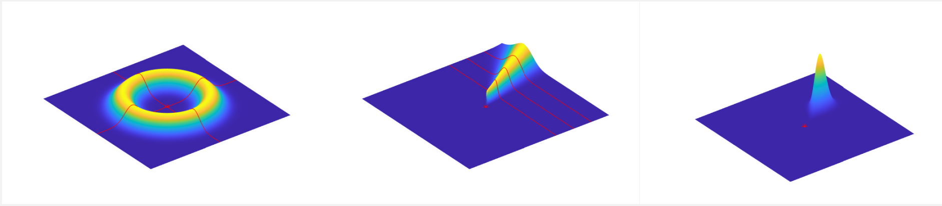
Robot localisation

We then measure the range and heading to the i-th beacon at known point $\left[z_1^{(i)}, z_2^{(i)}\right]^\mathrm{T}$.

$$y_1^{(i)}(t+1) = h_1(\boldsymbol{x}, \boldsymbol{v}) = \sqrt{\left(x_1 - z_1^{(i)}\right)^2 + \left(x_2 - z_2^{(i)}\right)^2} + v_1 \qquad \text{Range}$$

$$y_2^{(i)}(t+1) = h_2(\boldsymbol{x}, \boldsymbol{v}) = \arctan\left(\frac{z_2^{(i)} - x_2}{z_1^{(i)}}\right) - x_3 + v_2 \qquad \text{Bearing}$$

Where v_1 and v_2 are the noise in the range and bearing measurements respectively.



Extended Kalman Filters – General Form

Model Formation:

$$\begin{aligned} \boldsymbol{A}(t-1) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\big(\hat{\boldsymbol{x}}(t-1|t-1), \boldsymbol{u}(t-1)\big) \\ \boldsymbol{G}(t-1) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}\big(\hat{\boldsymbol{x}}(t-1|t-1), \boldsymbol{u}(t-1)\big) \\ \boldsymbol{C}(t) &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\big(\hat{\boldsymbol{x}}(t|t-1)\big) \\ \boldsymbol{H}(t) &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{v}}\big(\hat{\boldsymbol{x}}(t|t-1)\big) \end{aligned}$$

Prediction:

$$\hat{\boldsymbol{x}}(t|t-1) = \boldsymbol{f}\big(\hat{\boldsymbol{x}}(t-1|t-1), \boldsymbol{u}(t-1), 0\big) \tag{1}$$

$$\boldsymbol{P}(t|t-1) = \boldsymbol{A}\boldsymbol{P}(t-1|t-1)\boldsymbol{A}^{\mathsf{T}} + \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^{\mathsf{T}} \tag{2}$$

Correction:

$$\boldsymbol{L}(t) = \boldsymbol{P}(t|t-1)\boldsymbol{C}^{\mathsf{T}}\left[\boldsymbol{C}\boldsymbol{P}(t|t-1)\boldsymbol{C}^{\mathsf{T}} + \boldsymbol{H}\boldsymbol{R}\boldsymbol{H}^{\mathsf{T}}\right]^{-1} \tag{3}$$

$$\begin{aligned} \hat{\boldsymbol{x}}(t|t) &= \hat{\boldsymbol{x}}(t|t-1) + \boldsymbol{L}\big(\boldsymbol{y}(t) - \boldsymbol{h}(\hat{\boldsymbol{x}}(t|t-1))\big) \\ \boldsymbol{P}(t|t) &= (\boldsymbol{I} - \boldsymbol{L}\boldsymbol{C})\,\boldsymbol{P}(t|t-1) \end{aligned}$$

EKF formulation of robot motion equations

$$\boldsymbol{f} = \begin{bmatrix} x_1 + u_1 \cos x_3 \\ x_2 + u_1 \sin x_3 \\ x_3 + u_2 \end{bmatrix} + \begin{bmatrix} w_1 \cos x_3 \\ w_1 \sin x_3 \\ w_2 \end{bmatrix}$$

$$\rightsquigarrow \boldsymbol{A} := \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}} = \begin{bmatrix} 1 & 0 & -u_1 \sin \hat{x}_3 \\ 0 & 1 & u_1 \cos \hat{x}_3 \\ 0 & 0 & 1 \end{bmatrix}$$

Notice that we don't need to recalculate the $\boldsymbol{A}$ matrix at each time step. We can simply substitute what we think the current heading is, as well as the u_1 command that we intend to apply.

Note that we will only use this linearisation to calculate our covariance matrix. It is for this reason that we use the estimate of the robot's pose ($\hat{\boldsymbol{x}}$) as the operating point, rather than the true state $\boldsymbol{x}$.

EKF formulation of robot motion equations

We can similarly calculate $\boldsymbol{G}$.

$$\boldsymbol{f} = \begin{bmatrix} x_1 + u_1 \cos x_3 \\ x_2 + u_1 \sin x_3 \\ x_3 + u_2 \end{bmatrix} + \begin{bmatrix} w_1 \cos x_3 \\ w_1 \sin x_3 \\ w_2 \end{bmatrix}$$

$$\rightsquigarrow \boldsymbol{G} := \left. \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}} \right|_{\hat{\boldsymbol{x}}} = \begin{bmatrix} \cos \hat{x}_3 & 0 \\ \sin \hat{x}_3 & 0 \\ 0 & 1 \end{bmatrix}$$

Noise model for robot motion noise

It is often assumed that the noise variance in the motion of a robot is proportional to how fast it is moving. This makes some intuitive sense if you think of the noise arising from factors such as slip in the wheels. One particularly attractive feature of this idea is that there will be no noise when the robot is moving. To use this convention, we adopt a process noise covariance matrix of form

$$\boldsymbol{Q} = \begin{bmatrix} u_1^2 q_1 & 0 \\ 0 & u_2^2 q_2 \end{bmatrix}$$

where q_1 and q_2 are constants that describe the relationship between desired motion and the noise produced.

Sometimes using a more complex noise model might be justified. For example, a nonzero correlation between forward motion and rotational noise might be posited, as a faster moving robot might have more tendency to steer away from its intended path.

EKF formulation of robot motion equations

Finally we can calculate $\boldsymbol{C}$ and $\boldsymbol{H}$ for measurements to the i-th beacon.

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{v}, i) = \begin{bmatrix} \sqrt{\left(x_1 - z_1^{(i)}\right)^2 + \left(x_2 - z_2^{(i)}\right)^2} + v_1 \\ \arctan\left(\frac{z_2^{(i)} - x_2}{z_1^{(i)} - x_1}\right) - x_3 + v_2 \end{bmatrix}$$

$$\rightsquigarrow \boldsymbol{C}^{(i)} := \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}} = \begin{bmatrix} \frac{\hat{x}_1 - z_1^{(i)}}{\sqrt{\left(\hat{x}_1 - z_1^{(i)}\right)^2 + \left(\hat{x}_2 - z_2^{(i)}\right)^2}} & \frac{\hat{x}_2 - z_2^{(i)}}{\sqrt{\left(\hat{x}_1 - z_1^{(i)}\right)^2 + \left(\hat{x}_2 - z_2^{(i)}\right)^2}} & 0 \\ \frac{-(\hat{x}_2 - z_2)}{1 + \left(\frac{\hat{x}_2 - z_2^{(i)}}{\hat{x}_1 - z_1^{(i)}}\right)^2 \left(\hat{x}_1 - z_1^{(i)}\right)^2} & \frac{1}{1 + \left(\frac{\hat{x}_2 - z_2^{(i)}}{\hat{x}_1 - z_1^{(i)}}\right)^2 \left(\hat{x}_1 - z_1^{(i)}\right)^2} & -1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\hat{x}_1 - z_1^{(i)}}{\rho_i} & \frac{\hat{x}_2 - z_2^{(i)}}{\rho_i} & 0 \\ \frac{-(\hat{x}_2 - z_2^{(i)})}{\rho_i^2} & \frac{\hat{x}_1 - z_1^{(i)}}{\rho_i^2} & -1 \end{bmatrix}$$

where ρ_i is the estimated range to the i-th beacon.

EKF formulation of robot motion equations

$$\text{and, } \boldsymbol{H} := \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{v}}\right|_{\hat{\boldsymbol{x}}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Noise model for sensors

In the range-bearing localisation problem we have two sensors, each having quite different error characteristics. It can be important to capture this in the $\boldsymbol{R}$ matrix.

The angle sensor is generally relatively simple, with a simple gaussian describing the spread of measurement error.

A range sensor can be more complicated, with the error being related to the range. Consequently we might adopt a measurement noise covariance with form

$$\boldsymbol{R} = \begin{bmatrix} \rho\sigma_{\text{range}}^2 & 0 \\ 0 & \sigma_{\text{heading}}^2 \end{bmatrix}$$

where ρ is the range to the target.

Occasionally it might be worth adopting a more complex noise model for the range noise, such as $k + \rho\sigma_{\text{range}}^2$ or similar.

EKF formulation of robot motion equations

In a practical application we will usually have multiple beacons to deal with, so the $\boldsymbol{C}$ matrix might not be as simple as indicated above. There are two ways that we can deal with multiple landmarks.

1. Process the measurements from each beacon one at a time, using a $\boldsymbol{C}$ matrix as above. That is, each measurement updates the estimate, which then becomes the prior for the next update.
2. Add two rows to the $\boldsymbol{C}$ matrix for each additional beacon that is added. For example, if we have k beacons,

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}^{(1)} \\ \boldsymbol{C}^{(2)} \\ \vdots \\ \boldsymbol{C}^{(k)} \end{bmatrix}$$

Localisation in practice

There are advantages to each of these approaches. One of the major issues in practical SLAM applications is that we can potentially have many beacons (possibly hundreds). At any given time we are unlikely to have measurements associated with each of them. Using a $\boldsymbol{C}$ matrix that contains measurements to all of the beacons at each time step will cause various computational issues.

- We don't want to use a non-zero $\boldsymbol{C}$ block if we didn't actually apply feedback via $\boldsymbol{h}$. When we use $\boldsymbol{C}$ to update our covariance we would be saying that we became more confident because we measured something, when in fact we did not.
- The $\boldsymbol{C}$ matrix would be bigger than it needs to be, so we would likely be doing more work in the computation than is necessary.

Localisation in practice

Real problems therefore normally contain a step where a new $\boldsymbol{C}$ matrix is constructed based on which beacons have produced measurements in a given time step.

Feedback to the Kalman filter estimate is then only applied from those measurements that actually exist.

We have used “beacon” to describe the targets of our measurements. This implies some sort of deliberately placed navigational aid that we are using. It can be that this is the case, but there are other options also. For example, we might use visual sensors, or radar, to detect salient *landmarks* that happen to be in the environment and use subsequent measurements to those for estimating our location. (Though in the localisation framework we would need to know where the landmarks are before we start.)

Data Association

Note that discussion above assumes that we always know where the beacons are, and which is which. This is sometimes true, but more often we need an extra stage to associate a measurement of a beacon with a particular expected beacon location.

This is generally done via validation gating. That is, we predict where we expect to see a beacon and if we get a measurement consistent with that location then we can assume we are detecting the expected beacon.

This will become even more of an issue if we don't know the beacon locations a-priori, but are trying to learn them along with our robot's location.

Part II

Simultaneous Localisation and Mapping

Simultaneous Localisation and Mapping

We will now extend our localisation process to deal with the situation where we do not know the locations of a set of beacons but must instead rely on landmarks. There is no real distinction between the two, but we have used a different term to highlight that we have assumed that we know the location of beacons, but not generally of landmarks.

Because we do not know the location of landmarks a-priori, we must extend our filter to estimate the location of each landmark along with the robot's location.

In this treatment we will assume that the landmarks are stationary features of the environment. Something like a person would make a poor landmark in this setting.

The extension is not conceptually hard, though the problem does become more computationally complex. The basic idea is that we will just add the locations of the landmarks to the state vector and estimate them along with the robot state.

State vector

In a SLAM problem our state vector is

$$\boldsymbol{x} := \begin{bmatrix} x_{\text{robot}} \\ y_{\text{robot}} \\ \theta_{\text{robot}} \\ x^{(1)} \\ y^{(1)} \\ x^{(2)} \\ y^{(2)} \\ \vdots \\ x^{(k)} \\ y^{(k)} \end{bmatrix}$$

Where $x^{(i)}$ and $y^{(i)}$ are the x and y positions of the i-th landmark.

Linearisation in the SLAM problem

The $\boldsymbol{A}$ matrix we derived in the localisation problem is still largely valid. When the robot moves the landmarks do not, so the model for updating them is trivial. we just insert an appropriately sized identity matrix into the $\boldsymbol{A}$ matrix to describe $x^{(i)}(t+1) = x^{(i)}(t)$ and similar.

$$\boldsymbol{A} := \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}} = \begin{bmatrix} 1 & 0 & -u_1 \sin \hat{x}_3 & 0 \\ 0 & 1 & u_1 \cos \hat{x}_3 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \boldsymbol{I}_{2k} \end{bmatrix}$$

Similarly, the noise in the robot motion does not affect the beacons, so we simply append rows of zeros to the $\boldsymbol{G}$ matrix.

$$\boldsymbol{G} = \begin{bmatrix} \cos \hat{x}_3 & 0 \\ \sin \hat{x}_3 & 0 \\ 0 & 1 \\ 0_{2k\times 2} & \end{bmatrix}$$

Linearisation in the SLAM problem

We have the same measurement equation as earlier, as shown. However, now each measurement is taken between two unknown points, rather than a single unknown point and a known beacon. This means that our derived $\boldsymbol{C}$ matrix will contain extra terms arising from the partial derivatives of the measurements with respect to the location of the corresponding landmark.

$$h(\boldsymbol{x}, i) = \begin{bmatrix} \sqrt{\left(x_1 - z_1^{(i)}\right)^2 + \left(x_2 - z_2^{(i)}\right)^2} + v_1 \\ \arctan\left(\frac{z_2^{(i)} - x_2}{z_1^{(i)} - x_1}\right) - x_3 + v_2 \end{bmatrix}$$

$$\rightsquigarrow \boldsymbol{C}^{(i)} = \begin{bmatrix} \frac{\hat{x}_1 - z_1^{(i)}}{\rho_i} & \frac{\hat{x}_2 - z_2^{(i)}}{\rho_i} & 0 & \cdots & \frac{-\hat{x}_1 + z_1^{(i)}}{\rho_i} & \frac{-\hat{x}_2 + z_2^{(i)}}{\rho_i} & \cdots \\ \frac{-\hat{x}_2 + z_2^{(i)}}{\rho_i^2} & \frac{\hat{x}_1 - z_1^{(i)}}{\rho_i^2} & -1 & \cdots & \frac{\hat{x}_2 - z_2^{(i)}}{\rho_i^2} & \frac{\hat{x}_1 + z_1^{(i)}}{\rho_i^2} & \cdots \end{bmatrix}$$

The complete $\boldsymbol{C}$ matrix will have two rows of this form for each landmark. Most of the matrix will be zero, with only the first column and the diagonal being full.

$\boldsymbol{C}$ matrix in the two landmark case

For example, if there are two landmarks, then the $\boldsymbol{C}$ matrix becomes.

$$\boldsymbol{C} = \begin{bmatrix} \frac{\hat{x}_1 - z_1^{(1)}}{\rho_1} & \frac{\hat{x}_2 - z_2^{(1)}}{\rho_1} & 0 & \frac{-\hat{x}_1 + z_1^{(1)}}{\rho_1} & \frac{-\hat{x}_2 + z_2^{(1)}}{\rho_1} & 0 & 0 \\ \frac{-\hat{x}_2 + z_2^{(1)}}{\rho_1^2} & \frac{\hat{x}_1 - z_1^{(1)}}{\rho_1^2} & -1 & \frac{\hat{x}_2 - z_2^{(1)}}{\rho_1^2} & \frac{\hat{x}_1 + z_1^{(1)}}{\rho_1^2} & 0 & 0 \\ \frac{\hat{x}_1 - z_1^{(2)}}{\rho_2} & \frac{\hat{x}_2 - z_2^{(2)}}{\rho_2} & 0 & 0 & 0 & \frac{-\hat{x}_1 + z_1^{(2)}}{\rho_2} & \frac{-\hat{x}_2 + z_2^{(2)}}{\rho_2} \\ \frac{-\hat{x}_2 + z_2^{(2)}}{\rho_2^2} & \frac{\hat{x}_1 - z_1^{(2)}}{\rho_2^2} & -1 & 0 & 0 & \frac{\hat{x}_2 - z_2^{(2)}}{\rho_2^2} & \frac{\hat{x}_1 + z_1^{(2)}}{\rho_2^2} \end{bmatrix}$$

The form of this $\boldsymbol{C}$ matrix shows that each measurements now affects our estimate of the robot's position, and of the landmark's position. Notice that the feedback to the two positions has opposite signs; this is because we would need to move the two in opposite directions to counter any error in our estimate.

The proportion of the error that is resolved by moving each of them will depend on the relative certainty of our current estimates of their positions.

Initialising landmarks

During operation we need to manage the appearance (and often disappearance) of various landmarks. When we see a new landmark we must

1. Add two state variables to describe its location. This also requires enlarging the covariance matrix $\boldsymbol{P}$ so that we can keep track of how confident we are about the location of the point.
2. Provide an initial estimate of its location, as well a covariance for that initial estimate. Note that we cannot generally do this a-priori, as a putative landmark placed in the wrong place will almost certainly have a poor linearisation. Typically we put the first estimate at the location of the first measurement, and allow the first measurement to determine the covariance of that location.
3. Add two extra rows to the $\boldsymbol{C}$ matrix to describe measurements to the new landmark.

If we decide to remove a landmark then we must reverse all of those steps.

Initialising landmarks

Sometimes we might know things about landmarks that help us form an initial estimate of their location. For example,

- Landmarks might be known to lie in straight lines. For example our landmarks might be painted onto a wall.
- Landmarks might be known to be in some relation to other landmarks. For example, the left edge of an object is known to be 5 m from its right edge.
- Landmarks might be constrained to be in front of known existing opaque objects.

All of these sorts of information can be used to produce better initial estimates, and/or more fully describe the initial covariance of the location.

Data Association

The data association problem becomes very significant in SLAM applications. This is because we typically do not know where the landmarks are before we begin, so validation gating is more awkward in the initial phases.

Additional data about the various landmarks are often recorded so that they can be matched to later measurements. The precise nature of this process depends on the particular sensor, but it could be something like an average colour, or a vibrational spectrum, or an image of the surroundings.

Bearing-only SLAM

Both the localisation and SLAM problems above have assumed that we have access to both range and bearing measurements to each landmark. This might not always be the case, for example in visual SLAM we analyse images to extract interesting parts of the image to act as landmarks. In its most common form this produces only bearings for measurements.

Bearing-only localisation and SLAM work much as the complete range-bearing technique described above, but we must remove the first row from each of the $\boldsymbol{C}^{(i)}$ matrices that we have shown.

Bearing only SLAM is somewhat trickier to get working than range-bearing, simply because we have less information. It is also important to be somewhat clever about initialising a new landmark given that we typically do not have an initial estimate of range from the first measurement.