University of California San Diego

CSE 276A: HW3 (Kalman-Filter SLAM)

Varun Vupparige
November 11, 2022

1 Introduction

As robots move more towards autonomy space, there would be more emphasis on real time software systems, and thus this makes Simultaneous Localization and Mapping (SLAM) a critical problem in robotics. SLAM is the computational problem to simultaneously estimates a map of the environment and pose of a moving robot relative to that map, without any previous information external to the robot except for the observations of the robot.

Over decades, lot of different SLAM approaches have been researched and have already been in use in many applications such as self-driving cars, unmanned aerial vehicles and underwater surveillance vehicles, Techniques such as Particle Filer, Extended Kalman Filter and neural networks have been researched. Kalman filter is the closed form solution to the Bayesian filtering equations for the filtering model, where the motion and observation models are linear Gaussian. It can be effectively applied to locate the robot's pose as a Gaussian Distribution.

But since most physical systems are non-linear in nature, it is required to linearize the system to apply this technique. The Extended Kalman Filter uses to taylor series expansion to linearize the system around the mean of the probability distribution. In this project, we implement kalman filter based SLAM to localize and build the map of april tag landmarks.

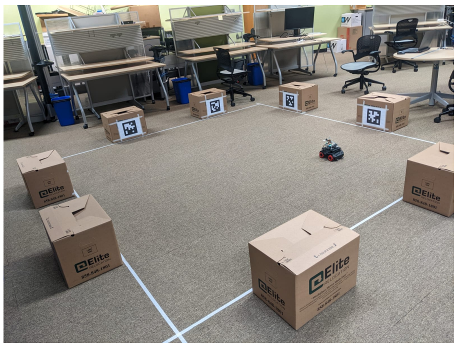

Figure 1.1: Environment Setip

2 Kalman Filter Algorithm

The Kalman Filter approach makes following assumptions to the SLAM problem:

- The prior probability density function $p_{t|t}$ is Gaussian

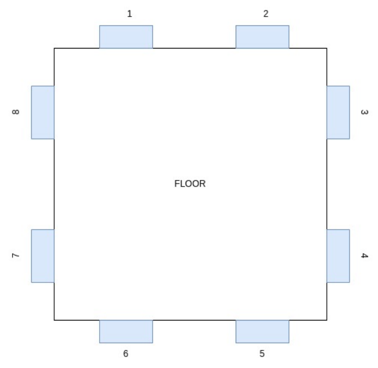

Figure 1.2: Top-down View

- The motion and observation models are affected by sensor noise which is also modelled as a Gaussian Distribution.
- The motion noise w_t and observation noise v_t are independent of each other, and state x_t

The motion model for a SLAM problem using Kalman Filter is given by:

$$x_{t+1} = f(x_t, u_t, w_t) = p_f(.|x_t, u_t) \tag{2.1}$$

$$\mathbf{w}_t \sim \mathcal{N}(0, W) \tag{2.2}$$

The observation model is defined as follows:

$$\mathbf{z}_t = h\left(\mathbf{x}_t, \mathbf{v}_t\right), \quad \mathbf{v}_t \sim \mathcal{N}(0, V) \tag{2.3}$$

Using the above defined Gaussian motion and observation model, the predicted and updated probability density function can be obtained using the Bayesian Filter equations as explained in the above subsection. But since the motion and observation models of most physical systems are non linear in nature, the predicted and updated pdf are not Gaussian. To address this issue, we make sure that the predicted and updated pdfs are Gaussian, by evaluating their first and second moments and approximating them with Gaussian with the same moments.

On applying moment matching and forcing the pdf to be Gaussian, the mean and co-variance of non-linear Kalman Filter prediction step is as follows:

$$\begin{aligned} \boldsymbol{\mu}_{t+1|t} &= \mathbb{E}\left[\mathbf{x}_{t+1} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t}\right] \\ &= \iint f\left(\mathbf{x}, \mathbf{u}_t, \mathbf{w}\right) \phi\left(\mathbf{x}; \boldsymbol{\mu}_{t|t}, \Sigma_{t|t}\right) \phi(\mathbf{w}; 0, W) d\mathbf{x} d\mathbf{w} \end{aligned} \tag{2.4}$$

$$\Sigma_{t+1|t} = \mathbb{E}\left[\left(\mathbf{x}_{t+1} - \mu_{t+1|t}\right)\left(\mathbf{x}_{t+1} - \mu_{t+1|t}\right)^\top \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t}\right]$$

$\iint f\left(\mathbf{x}, \mathbf{u}_t, \mathbf{w}\right) f\left(\mathbf{x}, \mathbf{u}_t, \mathbf{w}\right)^\top \phi\left(\mathbf{x}; \boldsymbol{\mu}_{t|t}, \Sigma_{t|t}\right) \phi(\mathbf{w}; 0, W) d\mathbf{x} d\mathbf{w} - \boldsymbol{\mu}_{t+1|t}\boldsymbol{\mu}_{t+1|t}^\top$(2.5) To evaluate the five integrals encountered in prediction and update steps as shown in eqn 8 and 9, the extended kalman filter uses a first order taylor series approximation to the motion and observation models around the state and noise. The motion model can be approximates as follows:

$$f\left(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t\right) \approx f\left(\boldsymbol{\mu}_{t|t}, \mathbf{u}_t, \mathbf{0}\right) + F_t\left(\mathbf{x}_t - \boldsymbol{\mu}_{t|t}\right) + Q_t \mathbf{w}_t \tag{2.6}$$

where,

$$F_t := \frac{df}{d\mathbf{x}}\left(\boldsymbol{\mu}_{t|t}, \mathbf{u}_t, \mathbf{0}\right) Q_t := \frac{df}{d\mathbf{w}}\left(\mu_{t|t}, \mathbf{u}_t, \mathbf{0}\right) \tag{2.7}$$

On evaluating the mean and co-variances of conditional Gaussian distribution, we get the following:

$$\begin{aligned}\boldsymbol{\mu}_{t+1|t} &= f\left(\boldsymbol{\mu}_{t|t}, \mathbf{u}_t, \mathbf{0}\right)\\ \Sigma_{t+1|t} &= F_t \Sigma_{t|t} F_t^\top + Q_t W Q_t^\top\end{aligned} \tag{2.8}$$

Similar to the motion model above, the observation model can be linearized using taylor series expansion as shown below:

$$h\left(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}\right) \approx h\left(\boldsymbol{\mu}_{t+1|t}, \mathbf{0}\right) + H_{t+1}\left(\mathbf{x}_{t+1} - \boldsymbol{\mu}_{t+1|t}\right) + R_{t+1}\mathbf{v}_{t+1} \tag{2.9}$$

where,

$$H_{t+1} := \frac{dh}{dx}\left(\boldsymbol{\mu}_{t+1|t}, \mathbf{0}\right) R_{t+1} := \frac{dh}{dv}\left(\boldsymbol{\mu}_{t+1|t}, \mathbf{0}\right) \tag{2.10}$$

On evaluating the mean and co-variances of conditional Gaussian distribution, we get the following:

$$\begin{aligned}\mu_{t+1|t+1} &= \boldsymbol{\mu}_{t+1|t} + K_{t+1|t}\left(z_{t+1} - h\left(\mu_{t+1|t}, 0\right)\right)\\ \Sigma_{t+1|t+1} &= \left(I - K_{t+1|t} H_{t+1}\right)\Sigma_{t+1|t}\end{aligned} \tag{2.11}$$

where,

$$K_{t+1|t} := \Sigma_{t+1|t} H_{t+1}^\top \left(H_{t+1}\Sigma_{t+1|t}H_{t+1}^\top + R_{t+1} V R_{t+1}^\top\right)^{-1} \tag{2.12}$$

2.1 Algorithm Implementation Details

The state vector s consists of the x, y and theta of the robot in the world frame. Each landmark is represented by its x and y positions. Thus the size of the state vector is 2*s+3.

In this assignment, we consider a range bearing observation model. Assume the robot can associate the observed asterisk (landmark) with its unique id, the sensor reading for the i-th asterisk at time step t is:

$$z_t^i = (r_t^i, \phi_t^i)^T$$

This leads to the prediction of the i-th asterisk position at time step t:

$$\begin{pmatrix}\bar{\mu}_{j,x}\\ \bar{\mu}_{j,y}\end{pmatrix} = \begin{pmatrix}\bar{\mu}_{t,x}\\ \bar{\mu}_{t,y}\end{pmatrix} + \begin{pmatrix}r_t^i\cos(\phi_t^i + \bar{\mu}_{t,\theta})\\ r_t^i\sin(\phi_t^i + \bar{\mu}_{t,\theta})\end{pmatrix}$$

The control vector u represents the velocities for the robot in its own frame. The elements of the vector are linear velocity in x, linear velocity in y and angular velocity at time t. The system matrix is simply F is the identity matrix of order n which represents the dimension of state vector s . This is because the state vector is already in the world frame, and we do not need any transformation. In our implementation, the control vector is already multiplied with the timestamp so G is a square matrix of zeros of size 2*n+3 with only the first three diagonal elements being equal to one. The measurement matrix H can be deduced easily from how we represent the measurements z using the state vectors as discussed above. It is a block matrix consisting of 3x3 identity matrices and zero matrices.

For the system noise, we only induce error in the robot motion starting at w = 0.1, and for the markers we ignore any noise since they do not move.

For the landmark detection, whenever the robot observes a new landmark for the first time, the state and covariance matrices and its measurement value are all updated by adding this specific marker with its unique ID and it is marked as True meaning that it is observed right now. However, if the observed measurements change after the prediction before executing the update step, we only use the of the markers that were observed before implying that any new marker observed will only be included in the next time step.

If any landmark that has been detected before goes out of the field of view, that landmark stays at the same index in the state matrix, however the value remains the same as it was when observed most recently and it is labelled as False meaning that it is not observed in the current timestep. So, for the Kalman update step, we take its prior value in . Whenever a previously detected landmark reappears, its value is simply updated at that index while labelling it as True before the Kalman update step. This is handled by the counter variable which can be referred in the submitted code.

3 Results

The size of the square and octagon trajectory was chosen such that the camera is able to observe maximum number of landmarks at a particular robot pose at time t. The results were only average since we needed more measurements to localize it better. Running the trajectory multiple times would improve the results considerably.

4 Results

4.1 Square Trajectory

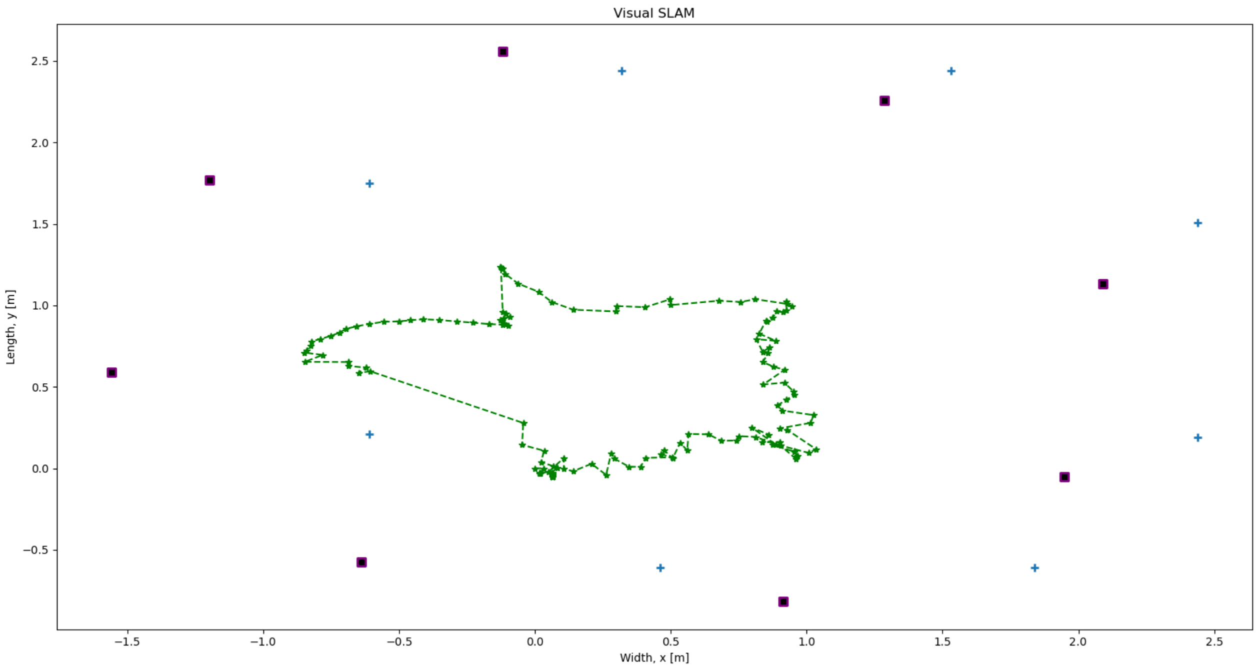

Figure 4.1: Square Trajectory

Error of Landmark 0: 0.7864098042207653
Error of Landmark 1: 0.4975676670163103
Error of Landmark 2: 0.5698443481168695
Error of Landmark 3: 1.0062374630963506
Error of Landmark 4: 1.375440820606778
Error of Landmark 5: 1.6515769803227653
Error of Landmark 7: 1.6575442646215548
Error of Landmark 8: 1.498827870418466
Average Error of Landmarks: 1.1304311523024826 m

4.2 Octagon Trajectory

5 Results

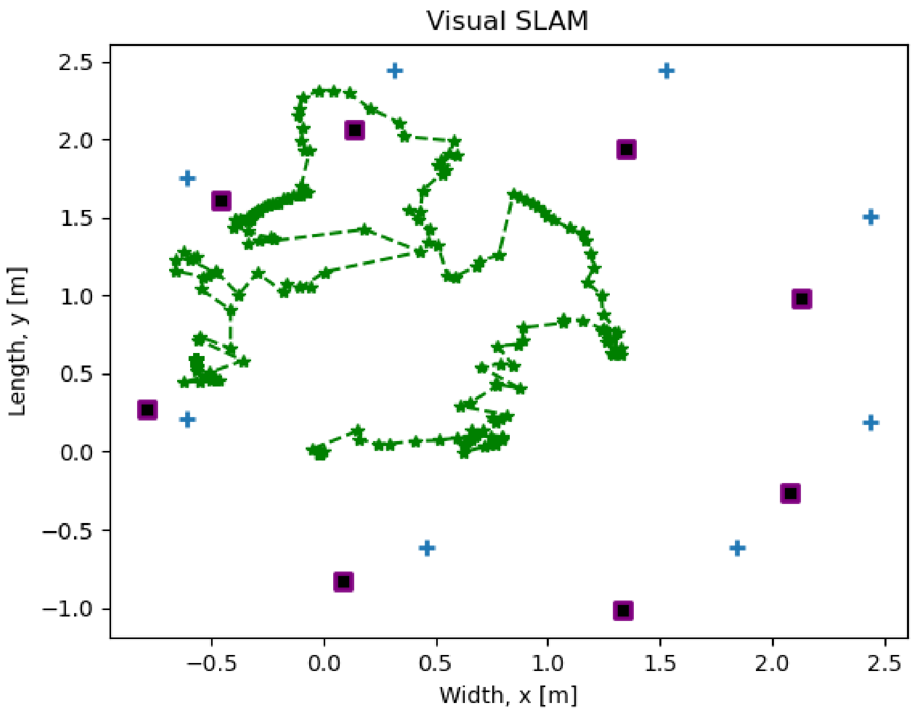

Figure 5.1: Square Trajectory

Error of Landmark 0: 0.18947706703926148
Error of Landmark 1: 0.435198365163817 05
Error of Landmark 2: 0.649872784322648
Error of Landmark 3: 0.5806247950070414
Error of Landmark 4: 0.6079964776605639
Error of Landmark 5: 0.5371187650573831
Error of Landmark 7: 0.4158017205169814
Error of Landmark 8: 0.2106844186335219
Average Error of Landmarks: 0.45334679917515225 m