State Estimation Localization Challenge

Manomit Bal
manomitb@buffalo.edu

1 Feature Based Localization

1.1 Problem Statement

This challenge provides a feature based localization engine with a computer simulated 2D soccer field, as inspired by the RoboCup competition. The objective of this challenge is to procure and accurately estimate the position and orientation $[x, y, \theta]^T$ of an autonomous mobile robot.

1.2 Features and World Description

1.2.1 Provided World Parameters

Markers and Indexes: There are provided four landmark markers and their corresponding indexes. Associated with these indexes are the corresponding x and y coordinates.

Range and Bearing: With each landmark is associated a corresponding observed distance and the observed bearing to the said landmark.

1.2.2 Provided Robot Parameters

Motion Deltas: The observed state differences $[\Delta x, \Delta y, \Delta\theta]^T$ from odometry in the form of local robot coordinates is provided from the localization controller.

Angle Perspective: The robot camera field of view angle in Degrees (**angle_fov**).

Sensor Noises: Provided are the sensor noise in range measurement (**sensor_noise_distance**) and the sensor noise in bearing measurement (**sensor_noise_orientation**).

Odometry Noises: Provided are the odometry noise with regard to
rotation-rotation (**odom_noise_rotation_from_rotation**),
rotation-translation (**odom_noise_rotation_from_translation**),
translation-rotation (**odom_noise_translation_from_rotation**)
and translation-translation (**odom_noise_translation_from_translation**).

2 System Design

When it comes to stochastic estimation, a number of methods can be utilized to acquire prominent state estimates with respect to the pose and orientation of a robot given its noisy sensor and odometry measurements. We utilize the **Bayes Filter** which is the most general algorithm for calculating belief distributions from measurement and control data.

The most straightforward application of the Bayes Filter to the localization problem is the **Markov Localization**. It helps transform a probabilistic belief at time t-1 to a belief at time t. This addresses the global localization problem, the position tracking problem and the kidnapped robot problem in static environments.

The most optimal implementation of Markov Localization is in the form on a **Kalman Filter**. It is essentially a parametrized set of mathematical equations that implement a prediction-update estimation which minimizes the estimated error covariance provided certain conditionals are met.

Kalman Filters work by making a prediction of the future, getting a measurement from reality, comparing the two, moderating this difference, and adjusting its estimate with this moderated value. They are also discrete and recursive. They rely on measurement samples taken between repeated but constant periods of time. The prediction of the future relies on the state of the present (position, velocity) as well as a guess a controllable model like a steering differential affecting the situation.

Kalman filters work optimally with linear stochastic difference equations. Thus in order to work with processes that are non linear, we require a filter that linearizes about the current mean and covariance. This is the **Extended Kalman Filter**.

2.0.1 Extended Kalman Filter

With known correspondences, we use a couple of linearized equations to estimate both the state and the covariance.

The following defines the algorithm and the variables used:

1. The first step is to extract the θ from the previous time step.

$\theta = \mu_{t-1,\theta}$

2. We build a state transition matrix A based and derived from the dynamics and kinematics of the differential drive robot. By multiplying the state by this and adding control factors, we get a prediction of the state for the next time step.

$$\mathrm{A} = \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

3. We build a Control Matrix which defines all the linear equations for all the control factors.

$$\mathrm{B} = \begin{pmatrix} \frac{-\sin\theta+\sin(\theta+\omega_t\Delta t)}{\omega_t} & \frac{v_t(\sin\theta-sin(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\cos(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ \frac{\cos\theta-\cos(\theta+\omega_t\Delta t)}{\omega_t} & -\frac{v_t(\cos\theta-cos(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\sin(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ 0 & \Delta t \end{pmatrix}$$

4. The following is the representation of the estimated process error covariance matrix at time t. α_1, α_2, α_3 and α_4 represent the odometry noises provided.

$$\mathrm{Q}_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2\omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4\omega_t^2 \end{pmatrix}$$

5. The state prediction and update equation consists of the addition of the control vector matrix which indicated the magnitude of control based on the dynamics of the system and the on the situation perceived at the particular time step.

$$\overline{\mu_t} = \mu_{t-1} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

6. The following represents the covariance prediction and update.

$$\overline{\Sigma}_t = A\Sigma_{t-1}A^T + BQB^T$$

7. The following represents the estimated measurement error covariance matrix that contains the sensor distance and bearing noise.

$$\mathrm{R} = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

8. We then loop through all possible landmarks perceived at a particular time.

For all observed features $\mathrm{z}_t^i = (r_t^i \quad \phi_t^i \quad s_t^i)^T do$

9. The distance squared between the landmark coordinates and the current robot coordinates at time t. The $m_{i,x}$ and $m_{i,y}$ represent the coordinates of the i^{th} landmark and $\overline{\mu}_{t,x}$ and $\overline{\mu}_{t,y}$ represents the coordinates of the robot at that time.

$\mathrm{q} = (\mathrm{m}_{i,x} - \overline{\mu}_{t,x})^2 + (m_{i,y} - \overline{\mu}_{t,y})^2$

10. The following is the estimated (calculated) matrix based on landmarks information, the distance and orientation.

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ atan2(m_{i,y} - \overline{\mu}_{t,y}, m_{i,x} - \overline{\mu}_{t,x}) - \overline{\mu}_{t,\theta} \end{pmatrix}$$

11. We build the observation matrix. We can multiply this with the state vector to translate it to a measurement vector.

$$\mathrm{H}_t^i = \begin{pmatrix} -\frac{m_{i,x} - \overline{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{i,y} - \overline{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \\ \frac{m_{i,y} - \overline{\mu}_{t,y}}{q} & \frac{m_{i,x} - \overline{\mu}_{t,x}}{q} & -1 \end{pmatrix}$$

12. The following is the innovation covariance matrix where we compare real error against prediction.

$\mathrm{S}_t^i = H_t^i \overline{\Sigma}_t [H_t^i]^T + Q_t$

13. We extract the Kalman Gain where we moderate the prediction.

$\mathrm{K}_t^i = \overline{\Sigma}_t [H_t^i]^T [S_t^i]^{-1}$

14. We procure the new estimate of where the robot is.

$\overline{\mu}_t = \overline{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$

15. We procure the new estimate of error.

$\overline{\Sigma}_t = (I - K_t^i H_t^i) \overline{\Sigma}_t$

end For

16. Updating both state and error estimates.

$\mu_t = \overline{\mu}_t$

$\Sigma_t = \overline{\Sigma}_t$

return μ_t, Σ_t

According to the problem statement, only the difference in range (Δx and the difference in orientation ($\Delta\theta$) is provided but not the time difference (Δ t). In order to use the existing equations we modify variables A, B and Q to accommodate the characterisation of the change in range and orientation without the change in time factor. A change in variables by substitution reveals the following new equations:

$$\mathrm{A} = \begin{pmatrix} 1 & 0 & -\frac{\Delta x}{\Delta\theta}\cos\theta + \frac{\Delta x}{\Delta\theta}\cos(\theta+\Delta\theta) \\ 0 & 1 & -\frac{\Delta x}{\Delta\theta}\sin\theta + \frac{\Delta x}{\Delta\theta}\sin(\theta+\Delta\theta) \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathrm{B} = \begin{pmatrix} -\sin\theta + \sin(\theta+\Delta\theta) & \frac{\Delta x}{\Delta\theta}(\sin\theta - \sin(\theta+\Delta\theta)) + \Delta x\cos(\theta+\Delta\theta) \\ \cos\theta - \cos(\theta+\Delta\theta) & -\frac{\Delta x}{\Delta\theta}(\cos\theta - \cos(\theta+\Delta\theta)) + \Delta x\sin(\theta+\Delta\theta) \\ 0 & \Delta\theta \end{pmatrix}$$

$$\mathrm{Q}_t = \begin{pmatrix} \alpha_1(\frac{\Delta x}{\Delta\theta})^2 + \alpha_2 & 0 \\ 0 & \alpha_3(\frac{\Delta x}{\Delta\theta})^2 + \alpha_4 \end{pmatrix}$$

These can now be used within the formulation to extract the appropriate state estimate of the robot with the known correspondences.

3 Visual Representation

The uncertainty of the robot pose is visualized with the help of a point cloud ellipse that represents the Gaussian of uncertainty surrounding the pose estimate in both the x and y domain. As the robot drifts without perceiving any landmark, the uncertainty increases, hence the ellipse increases. As soon as it sees a landmark, the robot estimate quickly converges and the size of the uncertainty ellipse decreases. This functionality is toggled by the 'd' or "D" character on the keyboard. It has been switched on by default at the beginning of the program execution.

4 Conclusion

After multiple iterations and testing phases, the EKF provides good convergence. There are certain scenarios when the robot looks at landmark [0] at the upper left corner and landmark [2] at the lower left corner and the localization is skewed for a second before it converges. Given some more time, I would debug the angular components based on the quadrants causing this minor drift.

Additionally, there was no noise associated with the Δx and $\Delta\theta$ as provided. I implemented a Gaussian noise generator to better simulate the noise as provided with the motion of the robot. The normal operation can be resumed by commenting out this part of the code that is marked accordingly in main.cpp.

5 Instructions and Procedures

5.1 Requirements

C++ 11: The Makefile as provided has been accommodated to be compiled with C++ 11 standard, depicted by "-std=c++11" under CXXFLAGS. This is mainly required for the Gaussian noise generator used.

Eigen Library: Run the following commands to install the eigen library. The INCLUDE flag in the Makefile takes care of the library call during compiling.

sudo apt-get update
sudo apt-get install libeigen3-dev

This library mostly takes care of all the matrix functions including Inverse and Transpose.

Glut Library: The Glut library is primarily used for GUI and visualization functionality. If you have a graphics card, it is required to be linked under the LIBRARIES tag in the Makefile. In my case, "-L/usr/lib/nvidia-367" was added.

sudo apt-get install freeglut3-dev

5.2 Procedure

Run command "make" in the folder with the core files. To run the executable, type:
./localization_test <input_file>

5.3 Keyboard/Mouse Mappings

Enter Key: Pause/Unpause program.

Space Key: Enter/Exit randomization mode (automated goal selections).

's': Changes the camera scan mode [1-4] (front, sweep, cursor, fixed).

'x': Disables all vision, Useful for testing odometry-only estimation.

'r': Toggles display of robot/visual field of view on/off.

'd': Toggles display if uncertainty ellipse on/off.

Left/Right Arrow Key: Changes the robot orientation manually.

Left Click: Selects new goal position at mouse pointer location.

Right Click: Selects new robot position at mouse pointer location.