Investigation of EKF in “robot_localization” ROS package

By Thanacha Choopojcharoen

To estimate general motion of robots moving in 3D, an impressive version of “Extended Kalman Filter” has been implemented as parts of a package “robot_localization” by Thomas (Tom) Moore at Charles River Analytics Inc. You can listen to his talk on the package in the following link.

https://www.ros.org/news/2016/06/tom-moore-working-with-the-robot-localization-package.html

https://roscon.ros.org/2015/presentations/robot_localization.pdf

The package has been widely used and considered to be popular among robotic researchers and developers around the world. Although there is an associated paper published under his name (as well as Daniel Stouch), the mathematics model of the underlying system is only briefly discussed. For those who are interested in the paper, the name of the paper is “A Generalized Extended Kalman Filter Implementation for the Robot Operating System by Moore, T. , & Stouch, D.”. The only option left is to investigate the source code “ekf.cpp” and gather information as much as possible. This is the link to the latest code.

https://github.com/cra-ros-pkg/robot_localization/blob/noetic-devel/src/ekf.cpp

Our goal is to identify the implemented nonlinear prediction model as well as its corresponding Jacobian matrix. After 2 days of investigation, this is what I found.

State Variables

There are 15 states in total, 3 for position, 3 for orientation (in roll-pitch-yaw), 3 for linear velocity, 3 for angular velocity, and 3 for linear acceleration. Although all states in the code were individually written out, we will use vector representation to ease our calculation.

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{r} \\ \mathbf{v} \\ \boldsymbol{\omega} \\ \mathbf{a} \end{bmatrix}$$

where

$\mathbf{x} \in \mathbb{R}^{15}$ is the state vector, $\mathbf{p} \in \mathbb{R}^3$ represents the position, $\mathbf{r} \in \mathbb{R}^3$ represents the orientation, $\mathbf{v} \in \mathbb{R}^3$ represents body's linear velocity, $\boldsymbol{\omega} \in \mathbb{R}^3$ represents body's angular velocity, and $\mathbf{a} \in \mathbb{R}^3$ represents body's linear acceleration.

Prediction Model

According to the code, the prediction model is represented by the used of "Transfer function", not to be mistaken as transfer function in classical control theory. The use of the word "Transfer function" in the code is most likely improper since an actual transfer function is normally referred to a ratio between output and input in frequency domain. The "Transfer function" is, in fact, a state transition matrix in state estimation theory. By grouping various terms, we finally obtain the prediction model implemented in this code.

$$
\begin{aligned}
\mathbf{p}_{k+1} &= \mathbf{p}_k + \mathbf{R}(\mathbf{r}_k)\left(\mathbf{v}_k \Delta t + \frac{1}{2}\mathbf{a}_k \Delta t^2\right) \\
\mathbf{r}_{k+1} &= \mathbf{r}_k + \mathbf{J}(\mathbf{r}_k)\boldsymbol{\omega}_k \Delta t \\
\mathbf{v}_{k+1} &= \mathbf{v}_k + \mathbf{a}_k \Delta t \\
\boldsymbol{\omega}_{k+1} &= \boldsymbol{\omega}_k + (\mathbf{u}_k^{\alpha})\Delta t \\
\mathbf{a}_{k+1} &= \mathbf{a}_k
\end{aligned}
$$

where

$$
\mathbf{R}(\mathbf{r}) = \mathbf{R}_z(r_z)\mathbf{R}_y(r_y)\mathbf{R}_x(r_x)
$$

$$
\mathbf{R}(\mathbf{r}) = \begin{bmatrix} \hat{\mathbf{x}}_\mathbf{0}^\top \\ \hat{\mathbf{y}}_\mathbf{0}^\top \\ \hat{\mathbf{z}}_\mathbf{0}^\top \end{bmatrix} = \begin{bmatrix} c_z c_y & c_z s_y s_x - s_z c_x & c_z s_y c_x + s_z s_x \\ s_z c_y & s_z s_y s_x + c_z c_x & s_z s_y c_x - c_z s_x \\ -s_y & c_y s_x & c_y c_x \end{bmatrix}
$$

$$
\mathbf{J}(\mathbf{r}) = \begin{bmatrix} 1 & s_x t_y & c_x t_y \\ 0 & c_x & -s_x \\ 0 & \frac{s_x}{c_y} & \frac{c_x}{c_y} \end{bmatrix}
$$

$\mathbf{u}_k^{\alpha}$ is the control in angular acceleration, $\mathbf{u}_k^{a}$ is the control in linear acceleration, ϕ is the mode whether the control in linear acceleration is provided, and Δt is the time-step.

Note that the rate of change of roll-pitch-yaw does not equal to the angular velocity. This can be proven by adding angular velocities in succession and transform to the body's coordinate frame.

$$
\boldsymbol{\omega} = \mathbf{R}(\mathbf{r})^\top \left(\begin{bmatrix} 0 \\ 0 \\ \dot{r}_z \end{bmatrix} + \mathbf{R}_z(r_z) \begin{bmatrix} 0 \\ \dot{r}_y \\ 0 \end{bmatrix} + \mathbf{R}_z(r_z)\mathbf{R}_y(r_y) \begin{bmatrix} \dot{r}_x \\ 0 \\ 0 \end{bmatrix} \right)
$$

$$
\boldsymbol{\omega} = \left(\mathbf{R}_z(r_z)\mathbf{R}_y(r_y)\mathbf{R}_x(r_x)\right)^\top \begin{bmatrix} 0 \\ 0 \\ \dot{r}_z \end{bmatrix} + \left(\mathbf{R}_y(r_y)\mathbf{R}_x(r_x)\right)^\top \begin{bmatrix} 0 \\ \dot{r}_y \\ 0 \end{bmatrix} + \mathbf{R}_x(r_x)^\top \begin{bmatrix} \dot{r}_x \\ 0 \\ 0 \end{bmatrix}
$$

$$\boldsymbol{\omega} = \begin{bmatrix}1\\0\\0\end{bmatrix}\dot{r}_x + \begin{bmatrix}-s_y\\c_y s_x\\c_y c_x\end{bmatrix}\dot{r}_z + \begin{bmatrix}0\\c_x\\-s_x\end{bmatrix}\dot{r}_y$$

$$\boldsymbol{\omega} = \begin{bmatrix}1 & 0 & -s_y\\0 & c_x & c_y s_x\\0 & -s_x & c_y c_x\end{bmatrix}\dot{\mathbf{r}}$$

By applying the inverse matrix, we finally obtain $\mathbf{J}(\mathbf{r})$.

Even though it seems like the author predicts the current states by multiplying the transition matrix to the previous states, some states are separately predicted.

The prediction model can be expressed in matrix equation as follows.

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$$

$$\begin{bmatrix}\mathbf{p}_{k+1}\\\mathbf{r}_{k+1}\\\mathbf{v}_{k+1}\\\boldsymbol{\omega}_{k+1}\\\mathbf{a}_{k+1}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0} & \mathbf{R}(\mathbf{r}_k)\Delta t & \mathbf{0} & \frac{1}{2}\mathbf{R}(\mathbf{r}_k)\Delta t^2\\\mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{J}(\mathbf{r}_k)\Delta t & \mathbf{0}\\\mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \Delta t\mathbf{I}_3\\\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0}\\\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3\end{bmatrix}\begin{bmatrix}\mathbf{p}_k\\\mathbf{r}_k\\\mathbf{v}_k\\\boldsymbol{\omega}_k\\\mathbf{a}_k\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\\mathbf{0}\\\mathbf{0}\\\mathbf{u}_k^{\alpha}\Delta t\\\mathbf{0}\end{bmatrix}$$

The validity of having such prediction model will not be discussed. So far, no mathematics mistake has been made in the code.

Jacobian Matrix

Jacobian matrix is required for updating the state's covariance matrix. If the prediction model were linear, the Jacobian matrix would be the state transition matrix. For our calculation, it might be more convenient to compute Jacobian matrix of each set of states based on its corresponding dependent variables.

The first set of state variables is the position $\mathbf{p}_{k+1}$. Its dynamics depends on, $\mathbf{p}_k$, $\mathbf{r}_k$, $\mathbf{v}_k$, and $\mathbf{a}_k$.

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{R}(\mathbf{r}_k)\left(\mathbf{v}_k\Delta t + \frac{1}{2}\mathbf{a}_k\Delta t^2\right) = \mathbf{f}_{\mathbf{p}}(\,\mathbf{p}_k, \mathbf{r}_k, \mathbf{v}_k, \mathbf{a}_k)$$

Some mathematic manipulation is done to obtain an explicit expression for its Jacobian matrix.

$$\frac{\partial \mathbf{f}_{\mathbf{p}}}{\partial \mathbf{r}_k} = \frac{\partial}{\partial \mathbf{r}_k}(\mathbf{p}_k + \mathbf{R}(\mathbf{r}_k)\mathbf{b}_k) = \begin{bmatrix}\mathbf{b}_k^{\top}\frac{\partial}{\partial \mathbf{r}_k}\left(\hat{\mathbf{x}}_0(\mathbf{r}_k)\right)\\\mathbf{b}_k^{\top}\frac{\partial}{\partial \mathbf{r}_k}\left(\hat{\mathbf{y}}_0(\mathbf{r}_k)\right)\\\mathbf{b}_k^{\top}\frac{\partial}{\partial \mathbf{r}_k}\left(\hat{\mathbf{z}}_0(\mathbf{r}_k)\right)\end{bmatrix} = \mathbf{L}(\mathbf{r}_k, \mathbf{v}_k, \mathbf{a}_k)$$

where

$$\mathbf{b}_k = \mathbf{v}_k \Delta t + \frac{1}{2}\mathbf{a}_k \Delta t^2$$

$$\frac{\partial}{\partial \mathbf{r}_k}\left(\hat{\mathbf{x}}_0(\mathbf{r}_k)\right) = \begin{bmatrix} 0 & -c_z s_y & -s_z c_y \\ c_z s_y c_x + s_z s_x & c_z c_y s_x & -s_z s_y s_x - c_z c_x \\ -c_z s_y s_x + s_z c_x & c_z c_y c_x & -s_z s_y c_x + c_z s_x \end{bmatrix}$$
$$\frac{\partial}{\partial \mathbf{r}_k}\left(\hat{\mathbf{y}}_0(\mathbf{r}_k)\right) = \begin{bmatrix} 0 & -s_z s_y & c_z c_y \\ s_z s_y c_x - c_z s_x & s_z c_y s_x & c_z s_y s_x - s_z c_x \\ -s_z s_y s_x - c_z c_x & s_z c_y c_x & c_z s_y c_x + s_z s_x \end{bmatrix}$$
$$\frac{\partial}{\partial \mathbf{r}_k}\left(\hat{\mathbf{z}}_0(\mathbf{r}_k)\right) = \begin{bmatrix} 0 & -c_y & 0 \\ c_y c_x & -s_y s_x & 0 \\ -c_y s_x & -s_y c_x & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{f_p}}{\partial \mathbf{p}_k} = \frac{\partial}{\partial \mathbf{p}_k}\left(\mathbf{p}_k + \mathbf{R}(\mathbf{r}_k)\left(\mathbf{v}_k \Delta t + \frac{1}{2}\mathbf{a}_k \Delta t^2\right)\right) = \mathbf{I}_3$$
$$\frac{\partial \mathbf{f_p}}{\partial \mathbf{v}_k} = \frac{\partial}{\partial \mathbf{v}_k}\left(\mathbf{p}_k + \mathbf{R}(\mathbf{r}_k)\left(\mathbf{v}_k \Delta t + \frac{1}{2}\mathbf{a}_k \Delta t^2\right)\right) = \mathbf{R}(\mathbf{r}_k)\Delta t$$
$$\frac{\partial \mathbf{f_p}}{\partial \mathbf{a}_k} = \frac{\partial}{\partial \mathbf{a}_k}\left(\mathbf{p}_k + \mathbf{R}(\mathbf{r}_k)\left(\mathbf{v}_k \Delta t + \frac{1}{2}\mathbf{a}_k \Delta t^2\right)\right) = \frac{1}{2}\mathbf{R}(\mathbf{r}_k)\Delta t^2$$

We can express the Jacobian matrix of the position with respect to the state vector.

$$\frac{\partial \mathbf{f_p}}{\partial \mathbf{x}_k} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{L}(\mathbf{r}_k, \mathbf{v}_k, \mathbf{a}_k) & \mathbf{R}(\mathbf{r}_k)\Delta t & 0 & \frac{1}{2}\mathbf{R}(\mathbf{r}_k)\Delta t^2 \end{bmatrix}$$

The next set of state variables is the orientation $\mathbf{r}_{k+1}$. Its dynamics depends on, $\mathbf{r}_k$, and $\boldsymbol{\omega}_k$.

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \mathbf{J}(\mathbf{r}_k)\boldsymbol{\omega}_k \Delta t = \mathbf{f_r}(\mathbf{r}_k, \boldsymbol{\omega}_k)$$

$$\frac{\partial \mathbf{f_r}}{\partial \mathbf{r}_k} = \frac{\partial}{\partial \mathbf{r}_k}(\mathbf{r}_k + \mathbf{J}(\mathbf{r}_k)\boldsymbol{\omega}_k \Delta t) = \mathbf{I}_3 + \begin{bmatrix} \boldsymbol{\omega}_k{}^{\top} \Delta t \frac{\partial}{\partial \mathbf{r}}([1 \quad s_x t_y \quad c_x t_y]^{\top}) \\ \boldsymbol{\omega}_k{}^{\top} \Delta t \frac{\partial}{\partial \mathbf{r}}([0 \quad c_x \quad -s_x]^{\top}) \\ \boldsymbol{\omega}_k{}^{\top} \Delta t \frac{\partial}{\partial \mathbf{r}}\left(\begin{bmatrix} 0 & \frac{s_x}{c_y} & \frac{c_x}{c_y} \end{bmatrix}^{\top}\right) \end{bmatrix} = \frac{\partial \mathbf{f_r}}{\partial \mathbf{r}_k} = \mathbf{I}_3 + \mathbf{M}(\mathbf{r}_k, \boldsymbol{\omega}_k)\Delta t$$

$$\mathbf{M}(\mathbf{r}_k, \boldsymbol{\omega}_k) = \begin{bmatrix} \boldsymbol{\omega}_k{}^{\top} \begin{bmatrix} 0 & 0 & 0 \\ c_x t_y & \frac{s_x}{c_y^2} & 0 \\ -s_x t_y & \frac{c_x}{c_y^2} & 0 \end{bmatrix} \\ \boldsymbol{\omega}_k{}^{\top} \begin{bmatrix} 0 & 0 & 0 \\ -s_x & 0 & 0 \\ -c_x & 0 & 0 \end{bmatrix} \\ \boldsymbol{\omega}_k{}^{\top} \begin{bmatrix} 0 & 0 & 0 \\ \frac{c_x}{c_y} & \frac{s_x t_y}{c_y} & 0 \\ -\frac{s_x}{c_y} & \frac{c_x t_y}{c_y} & 0 \end{bmatrix} \end{bmatrix} \Delta t$$

$$\frac{\partial \mathbf{f}_{\mathbf{r}}}{\partial \boldsymbol{\omega}_k} = \mathbf{J}(\mathbf{r}_k)\Delta t$$

We can express the Jacobian matrix of the orientation with respect to the state vector.

$$\frac{\partial \mathbf{f}_{\mathbf{r}}}{\partial \mathbf{x}_k} = [\mathbf{0} \quad \mathbf{I}_3 + \mathbf{M}(\mathbf{r}_k, \mathbf{v}_k, \mathbf{a}_k) \quad \mathbf{0} \quad \mathbf{J}(\mathbf{r}_k)\Delta t \quad \mathbf{0}]$$

The rest of the Jacobian matrices is the same as the state transition matrix. Therefore, we finally obtain the Jacobian matrix of the entire prediction model with respect to the state variables.

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}_k} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{L}(\mathbf{r}_k, \mathbf{v}_k, \mathbf{a}_k) & \mathbf{R}(\mathbf{r}_k)\Delta t & \mathbf{0} & \frac{1}{2}\mathbf{R}(\mathbf{r}_k)\Delta t^2 \\ \mathbf{0} & \mathbf{I}_3 + \mathbf{M}(\mathbf{r}_k, \mathbf{v}_k, \mathbf{a}_k) & \mathbf{0} & \mathbf{J}(\mathbf{r}_k)\Delta t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \Delta t \mathbf{I}_3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \end{bmatrix}$$

The output prediction model is very simple. The estimator relies on being given some, if not all, estimated state from other system. This mean that the output variables $\mathbf{z} \in \mathbb{R}^m$ is a combination of state variables **x.**

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k = \begin{bmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,n} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{m,2} & \delta_{m,2} & \cdots & \delta_{m,n} \end{bmatrix} \mathbf{x}_k$$

where

$$\delta_{i,j} \in \{0,1\}$$

$$\sum_i \delta_{i,j} = 1 \; \forall j = 1,2,\cdots,n$$

$$\sum_j \delta_{i,j} = 1 \; \forall i = 1,2,\cdots,m$$