CMPE245 Project 2

Chien-Pin Chen
06/07/2016

Part 1 Design my extended Kalman filter (EKF):
In Extended Kalman Filter, equation 1, 2, and 3 of the project 2 could rewrite to a full system model as:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \psi \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \{c\}\cdot g(\sin\psi\sin\varphi+\cos\psi\cos\varphi\sin\theta) \\ \{c\}\cdot g(\sin\psi\cos\varphi\sin\theta-\cos\psi\sin\varphi) \\ \{c\}\cdot(g\cos\varphi\cos\theta-g) \\ 0 \end{bmatrix} + \begin{bmatrix} & 0_{3\times4} & & \\ \delta_x & 0 & 0 & 0 \\ 0 & \delta_y & 0 & 0 \\ 0 & 0 & \delta_z & 0 \\ 0 & 0 & 0 & \delta_\psi \end{bmatrix}\begin{bmatrix} \xi_x \\ \xi_y \\ \xi_z \\ \xi_\psi \end{bmatrix} \tag{1}$$

And, equation 1 could be refer to:

$$\begin{aligned} dx &= xdt+\delta dw \\ X_{k+1} &= X_k + X_k\Delta t+\delta\sqrt{\Delta t}W \end{aligned} \tag{2}$$

Then, for computing, I could set:

$$\begin{aligned} x_{k+1} &= f_k(\hat{x}_k,0) \\ x_{k+1} &= \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \\ x_5(k+1) \\ x_6(k+1) \\ x_7(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k)+\Delta t\cdot x_4 \\ x_2(k)+\Delta t\cdot x_5 \\ x_3(k)+\Delta t\cdot x_6 \\ x_4(k)+\Delta t\cdot\{c\}\cdot g(\sin x_7\sin\varphi+\cos x_7\cos\varphi\sin\theta) \\ x_5(k)+\Delta t\cdot\{c\}\cdot g(\sin x_7\cos\varphi\sin\theta-\cos x_7\sin\varphi) \\ x_6(k)+\Delta t\cdot\{c\}\cdot(g\cos\varphi\cos\theta-g) \\ x_7(k) \end{bmatrix} \end{aligned} \tag{3}$$

$$\Phi_k = \frac{\partial f_k(\hat{x}_k,0)}{\partial x_k} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t\cdot\{c\}\cdot g(\cos x_7\sin\varphi+\sin x_7\cos\varphi\sin\theta) \\ 0 & 0 & 0 & 0 & 1 & 0 & \Delta t\cdot\{c\}\cdot g(\cos x_7\cos\varphi\sin\theta+\sin x_7\sin\varphi) \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\Gamma_k = \frac{\partial f_k(\hat{x}_k,0)}{\partial w_k} = \begin{bmatrix} & 0_{3\times4} & & \\ \delta_x\sqrt{dt} & 0 & 0 & 0 \\ 0 & \delta_y\sqrt{dt} & 0 & 0 \\ 0 & 0 & \delta_z\sqrt{dt} & 0 \\ 0 & 0 & 0 & \delta_\psi\sqrt{dt} \end{bmatrix} Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

In equation (3), (4), and (5), Δt and dt are non-constant sampling time which are assigned by dtrec in data3D.mat. Also, in equation (1), because $\xi_x(t)$, $\xi_y(t)$,

$\xi_z(t)$ and $\xi_\psi(t)$ are zero-mean white noises, I set δ_x, δ_y, δ_z and δ_ψ as one at first. I will tune those intensity of process noise according to the result of extended Kalman filter later. In addition, the parameter g is the gravitational acceleration (constant, as $9.81kgm/s^2$), and the parameter c is also a constant (set to 1) in part 1 of project 2. Next, the measurement model (equation (4) and (5)) of project 2 could refer to:

$$\underline{y}_k = \underline{h}_k(\underline{x}_k) + \underline{v}_k \tag{6}$$

then, for computing EKF, I could set:

$$\mathrm{H}_k = \frac{\partial h_k(\hat{x}_k, 0)}{\partial x_k} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$R = \begin{bmatrix} \mathrm{var}\{x_m\} & 0 & 0 & 0 \\ 0 & \mathrm{var}\{y_m\} & 0 & 0 \\ 0 & 0 & \mathrm{var}\{z_m\} & 0 \\ 0 & 0 & 0 & \mathrm{var}\{\psi_m\} \end{bmatrix} \tag{8}$$

In equation (8), $var\{x\}$, $var\{y\}$, and $var\{z\}$ are the variance of the measurement position and yaw angle of quad-rotor, and the measured noise of the Vicon is 10 cm. Therefore, I set $var\{x\}$, $var\{y\}$, and $var\{z\}$ as 0.1^2 for initialization. Also, in the provided video, the yaw angle of the quad-rotor did not change a lot, so I set $var\{\psi\} = 10\frac{\pi}{180}$ for initialization.
In the prediction step of Extended Kalman filter (EKF), I use following equations for discrete time model:

$$\hat{\underline{x}}_{k+1}(-) = \hat{\underline{f}}_k(\hat{\underline{x}}_k, 0) \tag{9}$$

$$\underline{P}_{k+1}(-) = \underline{\Phi}_k \underline{P}_k \underline{\Phi}_k^T + \underline{\Gamma}_k Q \underline{\Gamma}_k^T \tag{10}$$

For initial guess of $\hat{\underline{x}}_0(-)$, I use the first element of the measurement data as $E\{x(0)\}$, $E\{y(0)\}$, $E\{z(0)\}$, and $E\{yaw(0)\}$, and I also use the second element with first to compute initial velocity $E\{\dot{x}(0)\}$, $E\{\dot{y}(0)\}$, and $E\{\dot{z}(0)\}$:

$$x_0 = \begin{bmatrix} x_m(0) \\ y_m(0) \\ z_m(0) \\ x_m(1) - x_m(0)/\Delta t \\ y_m(1) - y_m(0)/\Delta t \\ z_m(1) - z_m(0)/\Delta t \\ yaw_m(0) \end{bmatrix} \tag{11}$$

For my initial guess of $\hat{\underline{P}}_0$, I use the variance of the measurement noise as initial values for the variance of process noise of the position (x, y, z, and yaw). Also, my initial expected velocity is acquired from initial position of quad-rotor, so I set the covariance of $\dot{x}_0$, $\dot{y}_0$, $\dot{z}_0$ according to the variance of the position. The following is how I initialize the state covariance matrix:

$$\begin{aligned}
&\mathrm{cov}\{x,x\} = P_{11} = P_{22} = P_{33} = measure\_noise^2 \\
&\mathrm{cov}\{\dot{x},\dot{x}\} = P_{44} = P_{55} = P_{66} = 2P_{11}/\Delta t^2 \\
&\mathrm{cov}\{\theta,\theta\} = P_{77} = (10^{pi}/180)^2 \\
&P_{14} = P_{25} = P_{36} = P_{11} \\
&P_{12} = P_{13} = P_{23} = P_{45} = P_{46} = P_{56} = 0 \\
&P_0 = \begin{bmatrix} P_{11} & 0 & 0 & P_{14} & 0 & 0 & 0 \\ 0 & P_{22} & 0 & 0 & P_{25} & 0 & 0 \\ 0 & 0 & P_{33} & 0 & 0 & P_{36} & 0 \\ P_{14} & 0 & 0 & P_{44} & 0 & 0 & 0 \\ 0 & P_{25} & 0 & 0 & P_{55} & 0 & 0 \\ 0 & 0 & P_{36} & 0 & 0 & P_{66} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P_{77} \end{bmatrix}
\end{aligned} \tag{12}$$

Next, I use following equations in the update step of EKF:

$$\underline{K}_{k+1} = \underline{P}_{k+1}(-)\underline{H}_{k+1}^T(\underline{H}_{k+1}\underline{P}_{k+1}(-)\underline{H}_{k+1}^T + \underline{R}_{k+1})^{-1} \tag{13}$$

$$\hat{\underline{x}}_{k+1} = \hat{\underline{x}}_{k+1}(-) + \underline{K}_{k+1}(\underline{y}_{k+1} - \underline{h}_{k+1}(\hat{\underline{x}}_{k+1}(-))) \tag{14}$$

$$\underline{P}_{k+1} = (I - \underline{K}_{k+1}\underline{H}_{k+1})\underline{P}_{k+1}(-) \tag{15}$$

After updating, do another extended Kalman filter with next measurement data $y(k)$, which is provided in data3d.mat file. In order to let the result of prediction to fit with the measure data (such as let the changing of x,y, and z to have the similar scale as the changing in the measurement data), I modify some values in the covariance matrices of the process and measured noise, as in figure 1 and figure 2. From the result of figure 1 (3D position), the predicted position is not fit with the measurement data, but the result of figure 2 shows that the predicted position (x, y, and z) did not diverge from the measurement data. Also, from the result of figure 2, the expected yaw (ψ) angle is more stable than the measurement yaw angle.

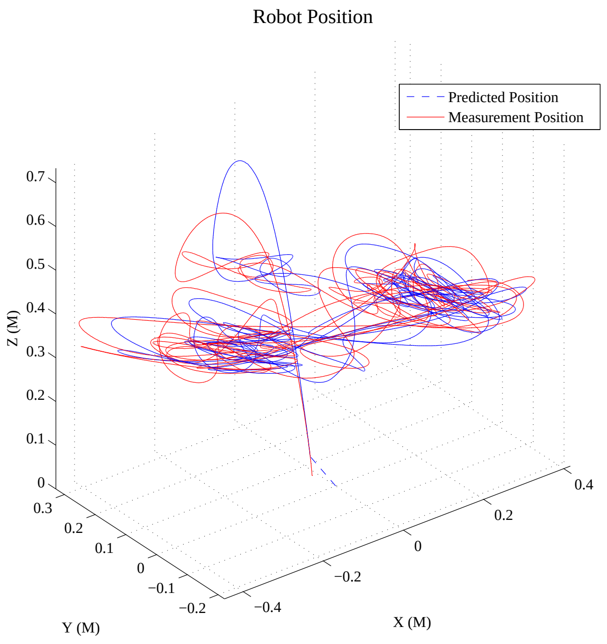

Figure 1: The result of Robot Position in 3D space.

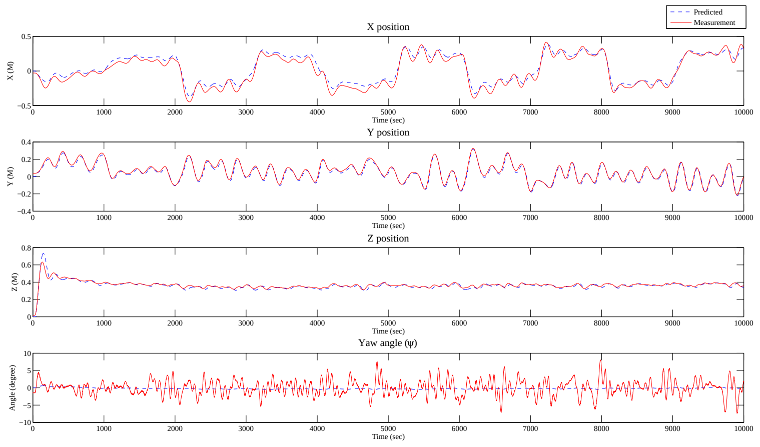

Figure 2: The result of Robot Position and orientation, separately.

Part 2 Find the best value for the parameter c_0
In part 1, c is a constant related to thrust, here I will use following equation to compute c:

$$c = (0.000409s^2 + 0.1405s - 0.099)/c_0 \ wheres = 255/600 \times (thrust) \tag{16}$$

In part 2, I change the values of c_0 (from 0.01 to $1.0e + 11$) and compute the average of residual (using equation (17)) to find out where I can get the minimum average residual, as in figure 3.

$$S(c_0) = \frac{1}{N}\sum_{i=1}^{N}(x_m(k) - x^-(k))^2 + (y_m(k) - y^-(k))^2 + (z_m(k) - z^-(k))^2 \tag{17}$$

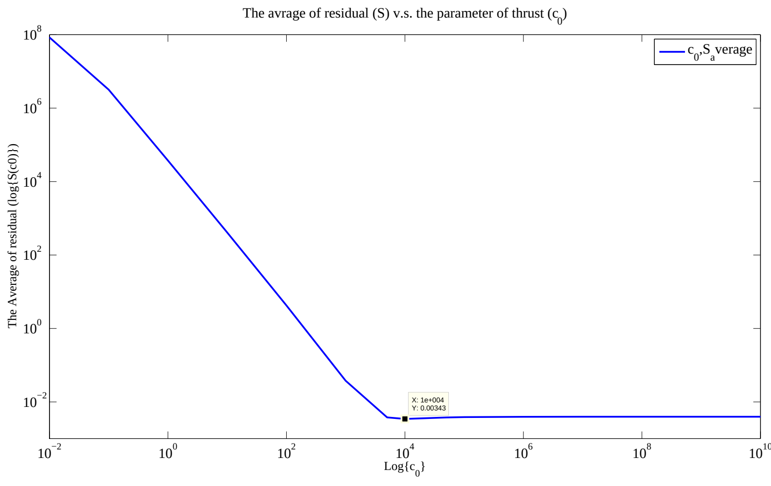

Figure 3: The average residual v.s. the parameter of thrust.

From the result of figure 3, when the parameter $c_0 = 10000$ (the best value), the extended Kalman filter will have the minimum average residual ($S(c_0) = 0.00343$)
Part 3 Deal with corrupt data with considering only x-y dynamics.
In part 3, because we only consider x-y dynamics, I will redesign my extended Kalman filter by remove parameter which is related to z and $\dot{z}$. Therefore, Φ is reduced to 5×5 matrix and Γ is reduced to 5×3 matrix. Next, because there are corrupted data of yaw angle in the given measurement data, I will use the same method in part 2 to get the average residual (498.16). Then, when compute the residual in the loop of Kalman filter, I will ignore the difference of yaw angle if the original residual is bigger than 498.16.

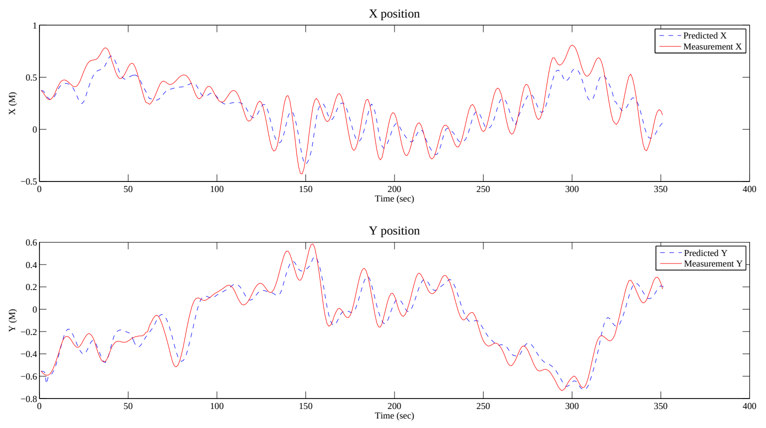

Figure 4: The result of Robot Position (x-y coordination).

From the result of figure 4, the expected robot position (x, y) are smooth and converge with the measurement data. Furthermore, in figure 5, the expected yaw angle is more smooth (with slightly shift) and follow the changing of the measurement yaw angle, and Kalman filter is able to detect those corrupted data from the given measurement data.

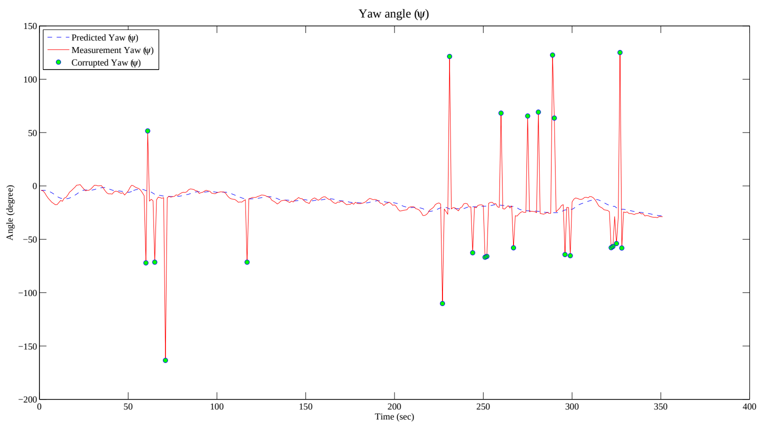

Figure 5: The result of yaw angle with corrupted data.