# Aided Navigation

Haidar Jamal

September 17, 2020

## 1 Introduction

In this report we derive an Extended Kalman Filter (EKF) that fuses data from an inertial measurement unit (IMU) along with wheel encoder data from a four wheeled skid steered rover. Our work is derived from [1] and [2], replacing the GPS measurement with wheel encoding. We use accelerometer and gyroscope data from the IMU. We approximate the rover model as a differential drive robot. We release a software implementation of this report in the references.

## 2 Notation

In our algorithm, we estimate the position, velocity, orientation, gyroscope bias, and accelerometer bias. We denote our state with $\boldsymbol{x}$:

$$\boldsymbol{x} = [p, v, b, x_g, x_a] \tag{1}$$

In the above, $p$ is position, $v$ is velocity, $b$ is the quaternion describing inertial to body frame transformation, $x_a$ is accelerometer bias, and $x_g$ is gyroscope bias.

An estimated variable will be denoted with a hat, such as $\hat{p}$. A sensor output with have a tilde above, such as $\tilde{u}$. The superscript $-$ denotes the estimate before a measurement update and $+$ denotes it after. The subscript $k$ denotes the time $t_k$, i.e, $P(t_k) = P_k$.

We assume our local frame is our inertial frame, and denote the inertial frame by $i$ and body frame by $b$. We assume the body frame aligns with the IMU output frame, and that the IMU frame is at the center of the rover.

## 3 Extended Kalman Filter

We use an error-state/feedback complementary filter EKF in our implementation. A block diagram overview of the filter is shown in Figure 1. In our implementation, the high rate sensor will be the IMU and the low rate sensor will be the predicted position and yaw from the wheel encoder model.

The algorithm is outlined in Figure 2 [1].

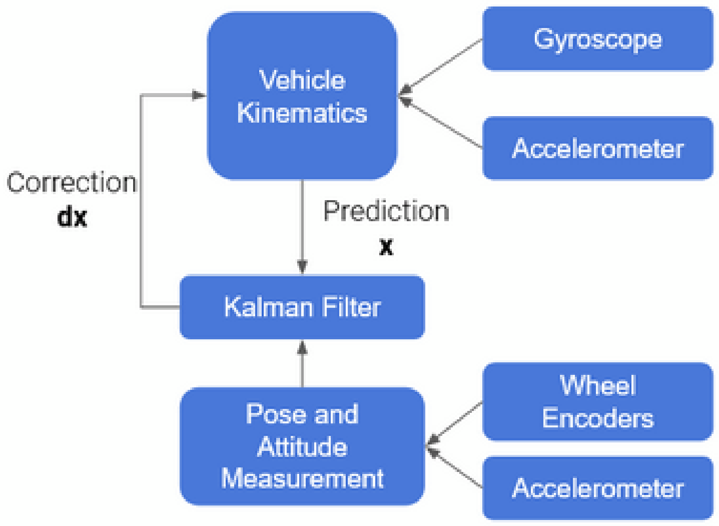

Figure 1: Feedback Complementary Filter Block Diagram

| | |
|---|---|
| Initialization | $\hat{\mathbf{x}}_0^- = \bar{\mathbf{x}}_0$ |
| | $\mathbf{P}_0^- = var(\delta\mathbf{x}_0^-)$ |
| Measurement Update | $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^\top\left(\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^\top\right)^{-1}$ |
| | $\mathbf{z}_k = \mathbf{y}_k - \bar{\mathbf{y}}_k$ |
| | $\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\mathbf{z}_k$ |
| | $\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_k^-$ |
| Time Propagation | $\dot{\hat{\mathbf{x}}}^-(t) = \mathbf{f}(\hat{\mathbf{x}}^-(t), \mathbf{u}(t), t), \;\text{for } t \in [t_k, t_{k+1})$ |
| | $\text{with } \hat{\mathbf{x}}^-(t_k) = \hat{\mathbf{x}}_k^+$ |
| | $\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k\mathbf{P}_k^+\mathbf{\Phi}_k^\top + \mathbf{Qd}_k$ |
| Definitions | $\mathbf{F}(\bar{\mathbf{x}}(t), \mathbf{u}(t), t) = \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\Big\|_{\mathbf{x}=\bar{\mathbf{x}}}$ |
| | $\bar{\mathbf{y}}_k = \bar{\mathbf{y}}(t_k) = \mathbf{h}(\hat{\mathbf{x}}^-(t_k), t_k)$ |
| | $\mathbf{H}_k = \mathbf{H}(\bar{\mathbf{x}}(t_k), t_k) = \frac{\partial\mathbf{h}}{\partial\mathbf{x}}\Big\|_{\mathbf{x}(t_k)=\bar{\mathbf{x}}(t_k)}$ |

Figure 2: Extended Kalman Filtering Algorithm

## 4 IMU Sensor Model

Let $u$ denote the gyroscope measurement. We denote $w_{ib}^b$ as the angular velocity of the body with respect to the inertial frame, in the body frame. Then, the gyroscope measurement model is

$$\tilde{w}_{ib}^b = w_{ib}^b + \Delta w_{ib}^b = w_{ib}^b + x_g + \nu_g \tag{2}$$

where $\nu_g$ is Gaussian white noise with PSD $\sigma_{\nu_g}^2$ and $x_g$ is a bias modeled as a first-order Gauss-Markov process:

$$\dot{x}_g = F_g x_g + w_g \tag{3}$$

where $F_g = -\lambda_g I$ and $w_g$ is a Gaussian white noise process with PSD $\sigma_{w_g}^2$.

The accelerometer measures the specific force vector $f^b$ in the body frame. We denote $G^b$ as the gravitational acceleration vector in the body frame. Furthermore, we denote $a_{ib}^b$ as the acceleration of the body with respect to the inertial frame, measured in the body frame. Then, the accelerometer measurement model is:

$$\tilde{f}^b = f^b + \Delta f^b = f^b + x_a + \nu_a = a_{ib}^b - G^b + x_a + \nu_a \tag{4}$$

where $\nu_a$ is Gaussian white noise with PSD $\sigma_{\nu_a}^2$ and $x_a$ is a bias modeled as a first-order Gauss-Markov process:

$$\dot{x}_a = F_a x_a + w_a \tag{5}$$

where $F_a = -\lambda_a I$ and $w_a$ is a Gaussian white noise process with PSD $\sigma_{w_a}^2$.

Thus, given measurements $\tilde{w}_{ib}^b$ and $\tilde{f}^b$, the angular rate and specific force vectors for use in the navigation equations are computed as

$$\hat{w}_{ib}^b = \tilde{w}_{ib}^b - \Delta\hat{w}_{ib}^b = \tilde{w}_{ib}^b - \hat{x}_g \tag{6}$$

$$\hat{f}^b = \tilde{f}^b - \Delta\hat{f}^b = \tilde{f}^b - \hat{x}_a \tag{7}$$

## 5 IMU Kinematic Model

In this section we derive the equations for computing the state estimate based on the integration of sensor signals through the vehicle kinematics. We denote the time between measurements as $\Delta t$.

The orientation update step uses the angular velocity measurement from the gyroscope to update orientation. The kinematic equation for the quaternion $b$ representing rotation from the $i$ frame to $b$ frame is

$$\dot{b} = \frac{1}{2}\begin{pmatrix} -b_2 & -b_3 & -b_4 \\ b_1 & b_4 & -b_3 \\ -b_4 & b_1 & b_2 \\ b_3 & -b_2 & -b_1 \end{pmatrix} w_{bi}^b$$

Thus, given an initial estimate of our orientation $\hat{b}(0)$ and $\hat{w}_{bi}^b$, we can integrate the above equation to compute $\hat{b}(t)$. In discrete time, we represent our orientation at time-step $k$ by $b(t_k)$, with $\Delta t$ as the time between time-steps.

From our sensor model, we have $\hat{w}_{ib}^{b} = \tilde{w}_{ib}^{b} - \hat{x}_g$, and since $\hat{w}_{ib}^{b} = -\hat{w}_{bi}^{b}$, we have

$$\hat{w}_{bi}^{b} = \hat{x}_g - \tilde{w}_{ib}^{b} \tag{8}$$

Then we can update our orientation estimate $\hat{b}(t_k)$ with the following equations [3]:

$$\tilde{b}_k = [cos\frac{\|w_{bi}^b\|\,\Delta t}{2}, sin\frac{\|w_{bi}^b\|\,\Delta t}{2}\frac{w_{bi}^b}{\|w_{bi}^b\|}] \tag{9}$$

$$\hat{b}(t_{k+1}) = \tilde{b}_k \hat{b}(t_k) \tag{10}$$

The velocity update step uses accelerometer data and integrates it between time-steps. Since the time derivative of velocity is acceleration,

$$\hat{v}_{ib}^{i}(t_{k+1}) = \hat{v}_{ib}^{i}(t_k) + \hat{a}_{ib}^{i}\Delta t \tag{11}$$

From our sensor model, we have $\tilde{f}^b = \hat{a}_{ib}^b - G^b + \hat{x}_a$. However, the above equation requires acceleration in the inertial frame, so we can rotate to the inertial frame and rearrange terms to obtain

$$\hat{a}_{ib}^{i} = \hat{R}_b^i(\tilde{f}^b - \hat{x}_a) + G^i \tag{12}$$

where $\hat{R}_b^i$ is computed by interpolating halfway between $\hat{b}(t_k)$ and $\hat{b}(t_{k+1})$ using the Slerp algorithm [4] and converting to direction cosine form. It is interpolated because accelerometer output values are generally averaged between measurement intervals. Further, $G^i = [0, 0, g]$, and $g$ is assumed to be the constant local acceleration due to gravity.

Since the time derivative of position is velocity, the position estimate is updated using

$$\hat{p}_{ib}^{i}(t_{k+1}) = \hat{p}_{ib}^{i}(t_k) + \hat{v}_{ib}^{i}(t_k)\Delta t + \hat{a}_{ib}^{i}\frac{\Delta t^2}{2} \tag{13}$$

## 6 EKF State Update

In this section a state space model for the navigation error vector is derived [1]. The general form of this model is

$$\Delta\dot{\boldsymbol{x}}(t) = F(t)\Delta\boldsymbol{x}(t) + G(t)\boldsymbol{w}(t) \tag{14}$$

The update is assumed to be performed at a fast enough rate such that all of the variables above are constant between time steps. Thus, $\Delta x$ is the state error vector, $w$ is the noise vector with covariance $Q$. From this model, we can compute the state transition matrix

$$\Phi_k \approx e^{F\Delta t} \tag{15}$$

The process noise covariance matrix $Qd$ at time step $k$ is approximated as

$$Qd_k = GQG^T\Delta t \tag{16}$$

The covariance at time step $k$ can be updated according to

$$P_{k+1} = \Phi_k P_k \Phi_k^T + Qd_k \tag{17}$$

We define the error vector as

$$\Delta\boldsymbol{x} = [\delta p, \delta v, \rho, \delta x_a, \delta x_g] \tag{18}$$

and the noise vector

$$\boldsymbol{w} = [\nu_a, \nu_g, w_a, w_g] \tag{19}$$

We drop superscripts denoting frames for clarity; unless specified the variable is assumed to be in the inertial frame. We describe each of these terms:

- $\delta p$ is the position error vector: $\delta p = p - \hat{p}$.
- $\delta v$ is the velocity error vector: $\delta v = v - \hat{v}$.
- $\rho$ is the orientation error: $\rho = [\epsilon_N, \epsilon_E, \epsilon_D]$. $\rho$ contains the small-angle rotations defined in the inertial frame that aligns the true orientation with the estimated orientation.
- $\delta x_a$ is the accelerometer bias error vector: $\delta x_a = x_a - \hat{x}_a$.
- $\delta x_g$ is the accelerometer bias error vector: $\delta x_g = x_g - \hat{x}_g$.

Also, substituting from the sensor models section, we compute

$$\delta f^b = \Delta f^b - \Delta\hat{f}^b = \delta x_a + \nu_a \tag{20}$$

$$\delta w_{ib}^b = \Delta w_{ib}^b - \Delta\hat{w}_{ib}^b = \delta x_g + \nu_g \tag{21}$$

Now we compute the elements of $F$ and $G$ for our system.

$$\delta\dot{p} = \dot{p} - \dot{\hat{p}} = v - \hat{v} = \delta v \tag{22}$$

$$\dot{\rho} = \hat{R}_b^i \delta w_{ib}^b = \hat{R}_b^i \delta x_g + \hat{R}_b^i \nu_g \tag{23}$$

$$\delta\dot{v} = \dot{v} - \dot{\hat{v}} = -[f^i \times]\rho + \hat{R}_b^i \delta f^v = -[f^i \times]\rho - \hat{R}_b^i \delta x_a - \hat{R}_b^i \nu_a \tag{24}$$

$$\delta\dot{x}_a = \dot{x}_a - \dot{\hat{x}}_a = F_a \delta x_a + w_a \tag{25}$$

$$\delta\dot{x}_g = \dot{x}_g - \dot{\hat{x}}_g = F_g \delta x_g + w_g \tag{26}$$

Putting the above in matrix form, where each element is 3x3, we have

$$F = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ 0 & 0 & -[\hat{f}^i \times] & -\hat{R}_b^i & 0 \\ 0 & 0 & 0 & 0 & \hat{R}_b^i \\ 0 & 0 & 0 & F_a & 0 \\ 0 & 0 & 0 & 0 & F_g \end{bmatrix} \tag{27}$$

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\hat{R}_b^n & 0 & 0 & 0 \\ 0 & \hat{R}_b^n & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \tag{28}$$

## 7 Rover Kinematic Model

In our EKF, the predicted position from the rover kinematic model serves as a measurement. The kinematic model for a differential drive robot moving in 3D is summarized in the following equations [1]:

$$\dot{x} = \frac{1}{2}(v_r + v_l)cos(\psi)cos(\theta_t) \tag{29}$$

$$\dot{y} = \frac{1}{2}(v_r + v_l)sin(\psi)cos(\theta_t) \tag{30}$$

$$\dot{z} = -\frac{1}{2}(v_r + v_l)sin(\theta_t) \tag{31}$$

where $\psi$, $\theta$, and $\phi$ are yaw, pitch, and roll, respectively. For a differential time $\Delta t$, we can approximate the following discrete system:

$$x_{t+1} \approx x_t + \dot{x}\Delta t = x_t + \frac{1}{2}(v_r + v_l)cos(\psi_t)cos(\theta_t)\Delta t \tag{32}$$

$$y_{t+1} \approx y_t + \dot{y}\Delta t = y_t + \frac{1}{2}(v_r + v_l)sin(\psi_t)cos(\theta_t)\Delta t \tag{33}$$

$$z_{t+1} \approx z_t + \dot{z}\Delta t = z_t - \frac{1}{2}(v_r + v_l)sin(\theta_t)\Delta t \tag{34}$$

Again, $v_l\Delta t = \Delta s_l$ and $v_r\Delta t = \Delta s_r$, where $\Delta s_l$ and $\Delta s_r$ are the distances travelled by each wheel during $\Delta t$. Substituting into the above equations, we have:

$$x_{t+1} = x_t + \dot{x}\Delta t = x_t + \frac{1}{2}(\Delta s_r + \Delta s_l)cos(\psi_t)cos(\theta_t) \tag{35}$$

$$y_{t+1} = y_t + \dot{y}\Delta t = y_t + \frac{1}{2}(\Delta s_r + \Delta s_l)sin(\psi_t)cos(\theta_t) \tag{36}$$

$$z_{t+1} = z_t + \dot{z}\Delta t = z_t - \frac{1}{2}(\Delta s_r + \Delta s_l)sin(\theta_t) \tag{37}$$

In matrix form our model is:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + \frac{1}{2}(\Delta s_r + \Delta s_l)cos(\psi_t)cos(\theta_t) \\ y_t + \frac{1}{2}(\Delta s_r + \Delta s_l)sin(\psi_t)cos(\theta_t) \\ z_t - \frac{1}{2}(\Delta s_r + \Delta s_l)sin(\theta_t) \end{bmatrix} \tag{38}$$

For the wheel encoders, a noise matrix can be written as: $U = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$, where $\sigma_l = k|\Delta s_1|$ and $\sigma_r = k|\Delta s_r|$ and k is some experimentally derived constant that describes wheel slip. Thus, we can compute a noise model $R$ by using the Jacobian $F_u$ with respective to wheel distances, $R = F_u U_t F_u^T$ evaluated at the mean of the posterior, where

$$F_u = \begin{bmatrix} \frac{1}{2}cos(\psi_t)cos(\theta_t) & \frac{1}{2}cos(\psi_t)cos(\theta_t) \\ \frac{1}{2}sin(\psi_t)cos(\theta_t) & \frac{1}{2}sin(\psi_t)cos(\theta_t) \\ -\frac{1}{2}sin(\theta_t) & -\frac{1}{2}sin(\theta_t) \end{bmatrix} \tag{39}$$

In our implementation, we used quaternions in place of Euler angles. The above is useful for derivation purposes.

## 8 EKF Measurement Model

We can summarize the above equations as a measurement model $\tilde{y} = h(\boldsymbol{x})$. Since the output of the position estimation is $[\hat{x}, \hat{y}, \hat{z}]$, we can define

$$H = \begin{bmatrix} I & 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \end{bmatrix} \tag{40}$$

At the end of the time propagation we have an estimate position $\hat{p}$ and orientation $\hat{b}$. Thus we can form a residual

$$z = \tilde{p} - \hat{p} \tag{41}$$

and use this in the EKF algorithm.

## 9 References

[1]: Farrell, J. (2008). Aided navigation: GPS with high rate sensors. McGraw-Hill, Inc..
[2]: Crassidis, J. L. (2006). Sigma-point Kalman filtering for integrated GPS and inertial navigation. IEEE Transactions on Aerospace and Electronic Systems, 42(2), 750-756.
[3]: Whitmore, S.A., Hughes, L.: Calif: Closed-form Integrator for the Quaternion (Euler Angle) Kinematics Equations, USpatent (2000)
[4]: Shoemake, Ken. "Animating Rotation with Quaternion Curves." ACM SIGGRAPH Computer Graphics Vol. 19, Issue 3, 1985, pp. 345–354.