Quaternion Kalman Filter for Rigid Body Pose Estimation

Benjamin Deming

Aug 13 2021

Problem Statement

An alien artifact is being tracked using markers (opti-track esq). The noisy marker measurements are relative to the world frame. The artifact has time invariant dynamics and is not under power. The know marker locations relative to the center of mass are shown in the following table:

Marker	X	Y	Z
0	-1.5	-3.0	-4.5
1	-1.5	-3.0	1.5
2	-1.5	1.0	-4.5
3	-1.5	1.0	1.5
4	0.5	-3.0	-4.5
5	0.5	-3.0	1.5
6	0.5	1.0	-4.5
7	0.5	1.0	1.5

The artifact is rotating as well as translating. The goal of this work is to design and implement an Extended Kalman Filter to track the pose of the artifact [1]. The state will be:

$$\mathbf{x} = [x\ y\ z\ v_x\ v_y\ v_z\ q_0\ q_1\ q_2\ q_3\ w_x\ w_y\ w_z]$$

I chose to represent the attitude of the artifact in terms of quaternoins in order to avoid gimbal lock which can be encountered when Euler angles are used. Gimbal lock occurs when pitch (careful with angle order/convention) is 90 or -90 [2]. Rotation matrices do not have the problem of gimbal lock, but they increase the computational complexity of the algorithm.

1 Process Model

First I will design the process model. The moment of inertial matrix in the body frame is given by:

$$I_{body} = \begin{bmatrix} 1.571428571428571 & 0 & 0 \\ 0 & 5.362637362637362 & 0 \\ 0 & 0 & 7.065934065934067 \end{bmatrix}$$

I use a constant velocity model for the position and linear velocity state predictions. For the quaternion prediction I exploit the relationship between the quaternion and the angular velocity [2] about the x, y, and z world frame axes. And lastly, for the angular velocity I use the following relationship between angular velocity and angular acceleration to predict the future angular velocity [3],[4]:

$$\dot{\omega} = I_w^{-1}[\omega \text{x} (I\omega)]$$

$$I_w = R * I_{body} * R$$

where R is the rotation matrix from the quaternion state estimate.

$$\begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \\ q_0 \\ q_1 \\ q_2 \\ q_3 \\ w_x \\ w_y \\ w_z \end{bmatrix}_{k+1} = f(x,u) = \begin{bmatrix} x + \Delta t v_x \\ y + \Delta t v_y \\ z + \Delta t v_z \\ v_x \\ v_y \\ v_z \\ q_0 + \frac{\Delta t}{2}[-q_1\omega_1 - q_2\omega_2 - q_3\omega_3] \\ q_1 + \frac{\Delta t}{2}[q_0\omega_1 - q_3\omega_2 + q_2\omega_3] \\ q_2 + \frac{\Delta t}{2}[q_3\omega_1 + q_0\omega_2 - q_1\omega_3] \\ q_3 + \frac{\Delta t}{2}[-q_2\omega_1 + q_1\omega_2 + q_0\omega_3] \\ \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \Delta t I_w^{-1}[(\omega \text{ x } (I_w \omega)] \end{bmatrix}_k$$

Note we need to normalize the quaternion vector. This is done after each predict and update step.
The f vector is non linear so an EKF must be used. I evaluate f at each predict step and use this to predict the future state. However I also need to compute the jacobian F in order to compute the predicted filter covariance. I find F using symbolic math toolbox in matlab.

2 Measurement Model

The sensor readings measure the position of each marker relative to the world frame. However, I am estimating the center of mass relative to the world frame. Thus my measurement model must map the COM estimate in world frame to marker location world frame. I start with the following equation:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \ldots \end{bmatrix} = \begin{bmatrix} x_{COM} \\ y_{COM} \\ z_{COM} \\ \ldots \end{bmatrix} - R \begin{bmatrix} x_{offset} \\ y_{offset} \\ z_{offset} \\ \ldots \end{bmatrix}$$

Where x_1, y_1, z_1 are the state estimates mapped to the marker 1 frame. x_{COM}, y_{COM}, z_{COM} are the state estimates of the position of the center of mass. x_{offset}, y_{offset}, z_{offset} are the given offset between the markers and the center of mass. R is the rotation matrix estimated as a state which rotates the artifact from the current orientation to the initial orientation (world frame). R is computed by converting the state estimate of the quaternion to a rotation matrix using the standard conversion listed in the following:

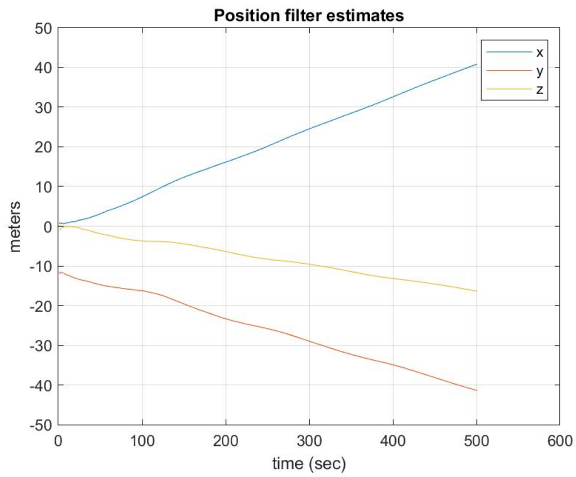
$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Thus the measurement model is formulated as:

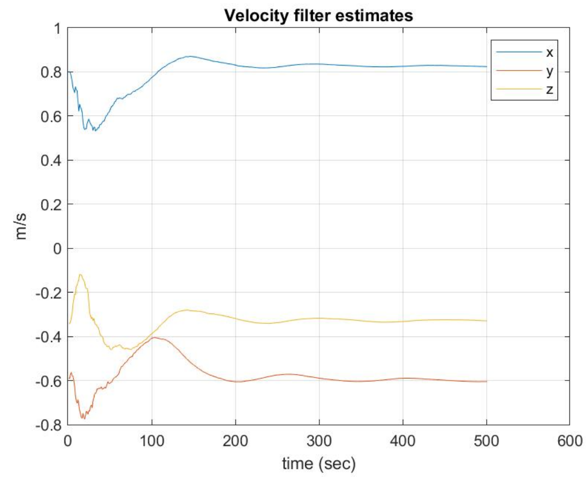
$$h(x) = \begin{bmatrix} x_{COM} - R_{11}x_{off} + R_{12}y_{off} + R_{13}z_{off} \\ y_{COM} - R_{21}x_{off} + R_{22}y_{off} + R_{23}z_{off} \\ z_{COM} - R_{31}x_{off} + R_{32}y_{off} + R_{33}z_{off} \\ \ldots \end{bmatrix}$$

The dimension of h(x) is 24 x 1 since we are mapping the state to the x, y, z coordinates of each of the 8 markers. Notice only the position and orientation from the state are used to map the state to the measurement frame. The rest of the states (linear and angular velocities) are hidden variables.
The measurement model is also nonlinear. I simply evaluate h at each update step and use it to compute the measurement residual. However I must compute the jacobian H at each update as well. H is used to compute the kalman gain and the filter covariance.

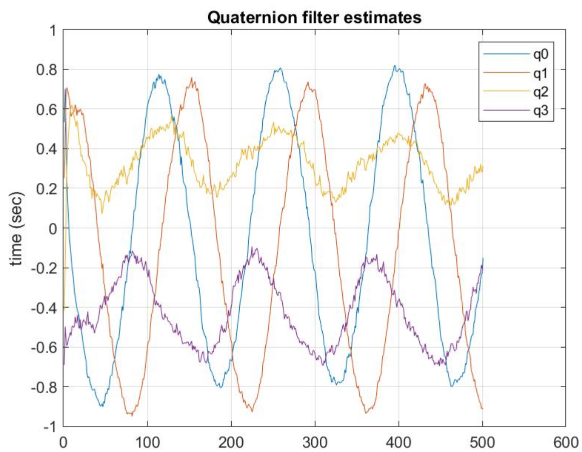
3 Results



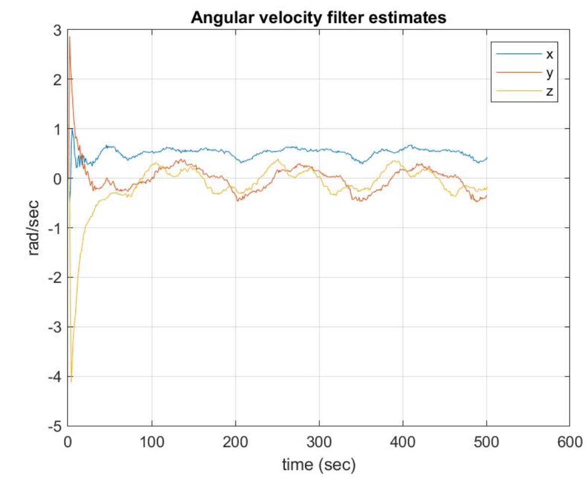
(a) Put your sub-caption here



(b) Put your sub-caption here

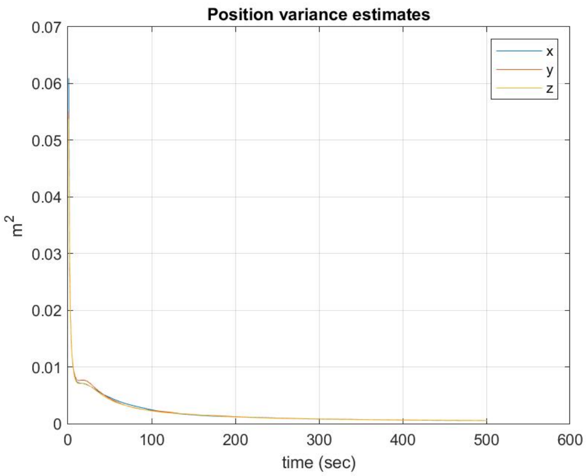


(c) Put your sub-caption here

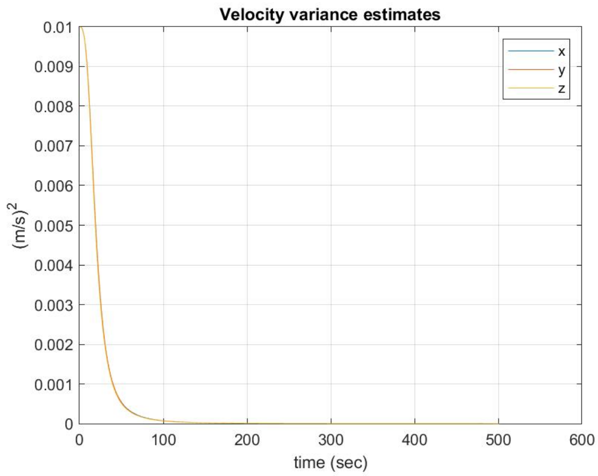


(d) Put your sub-caption here

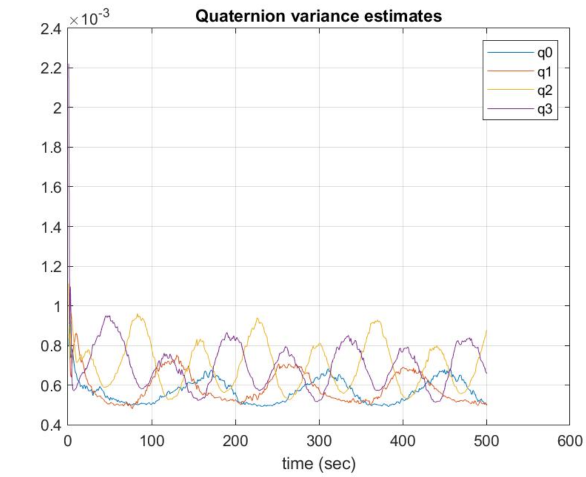
Figure 1: State estimates from the EKF



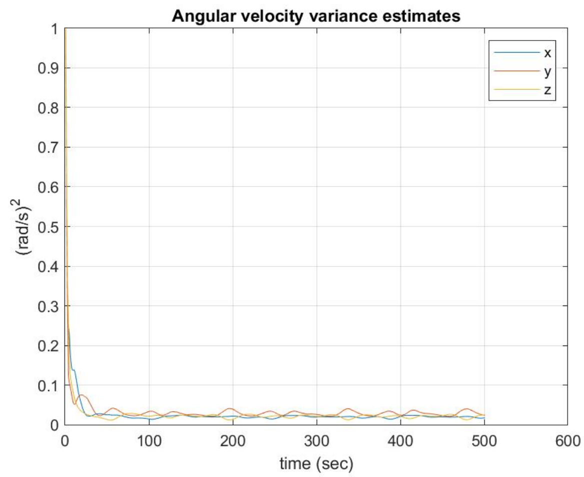
(a) Put your sub-caption here



(b) Put your sub-caption here



(c) Put your sub-caption here



(d) Put your sub-caption here

Figure 2: Variance estimates from the EKF

4 Filter Performance Evaluation

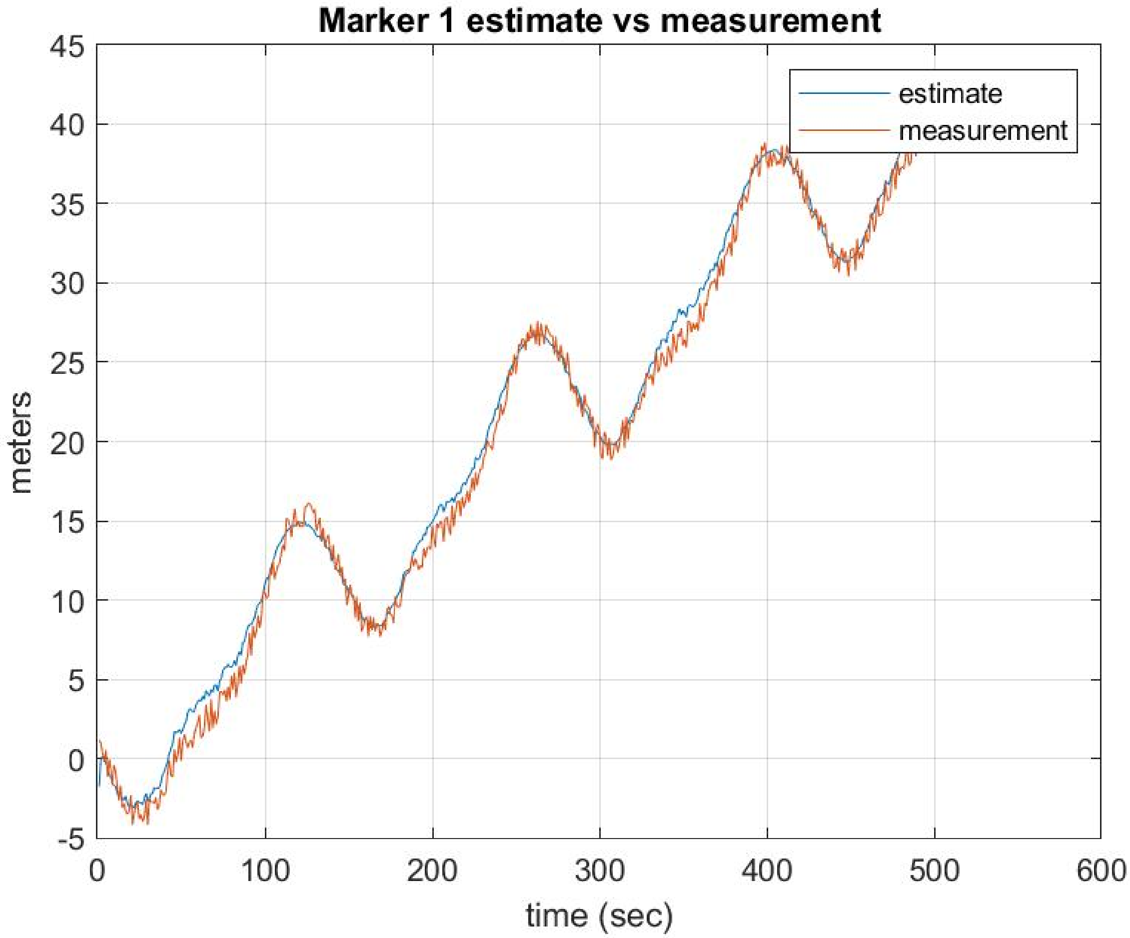


Figure 3: The measurement of the x axis of marker 1 vs the filter estimate mapped back to the marker frame. As shown in the plot, the filter estimate follows the x position of the measurement with little lag.

4.1 Measurement Residuals

The measurement residuals computed in the update step should be zero mean. Furthermore, in simulation, the difference between the ground truth and the filter estimate should stay between the variance (diagonal of P) of the filter.

4.2 Log Likelihood Function

This is important to us because we have the filter output and we want to know the likelihood that it is performing optimally given the assumptions of Gaussian noise and linear behavior.

4.3 Computational Efficiency

The computational efficiency of the filter can be improved in two ways. The first is that in both the predict and the update steps, I am using matlab's symbolic substitution to solve f, F, h, and H. This is computationally intensive. A better way would be to compute the functions and jacobians symbolically, and then make functions which take the filter states as inputs and output the evaluated jacobians. The only thing that needs to be done here is to hard code the functions with the process and measurement models.

The second way to improve computational efficiency is instead of linearizing the quaternion state, represent the small deviation from the nominal state as euler angles and represent the nominal state as quaternions [1]. Then apply the small angle approximations to greatly simplify the linearized rotation. Since you are just representing the small perturbation as euler angles and not the actual rotational state, you don't have to worry about encountering gimbal lock.

5 Next Steps

5.1 Fully Coupled 6 DOF Dynamics

$$\sum F = ma$$
$$0 = \text{ma}$$
$$\boxed{\text{a} = \alpha \times r + \omega \times (\omega \times r)}$$

And Euler's equations for the rotational dynamics:

$$\boxed{\tau = I\alpha + \omega \times I\omega = 0}$$

6 References

1. Problem Statement: https://www.cs.cmu.edu/ cga/kdc/ass4/

2. Quaternion EKF Tutorial: https://thepoorengineer.com/en/attitude-determination/

3. Rigid Body Dynamics EKF: https://www.hindawi.com/journals/mpe/2016/2962671/

4. Rigid Body Mechanics: https://arc.aiaa.org/doi/pdf/10.2514/4.867231

5. IMU/GPS EFK: https://openimu.readthedocs.io/en/latest/intro.html