Example EKF Implementation

Peter Kohler

April 16, 2023

Estimation Scenario:
This project's estimation scenario involves the use of an aruco marker. In the 3D simulator I am using, the turtlebot3 robot is equipped with an intel realsense camera. The digital images of that camera are being fed into an open source computer vision library (OpenCV) that is detecting the aruco marker and calculating the 3D coordinate transform from the camera's lens to the aruco marker. Figure 1 shows a visual representation of the detected aruco marker using the OpenCV library.

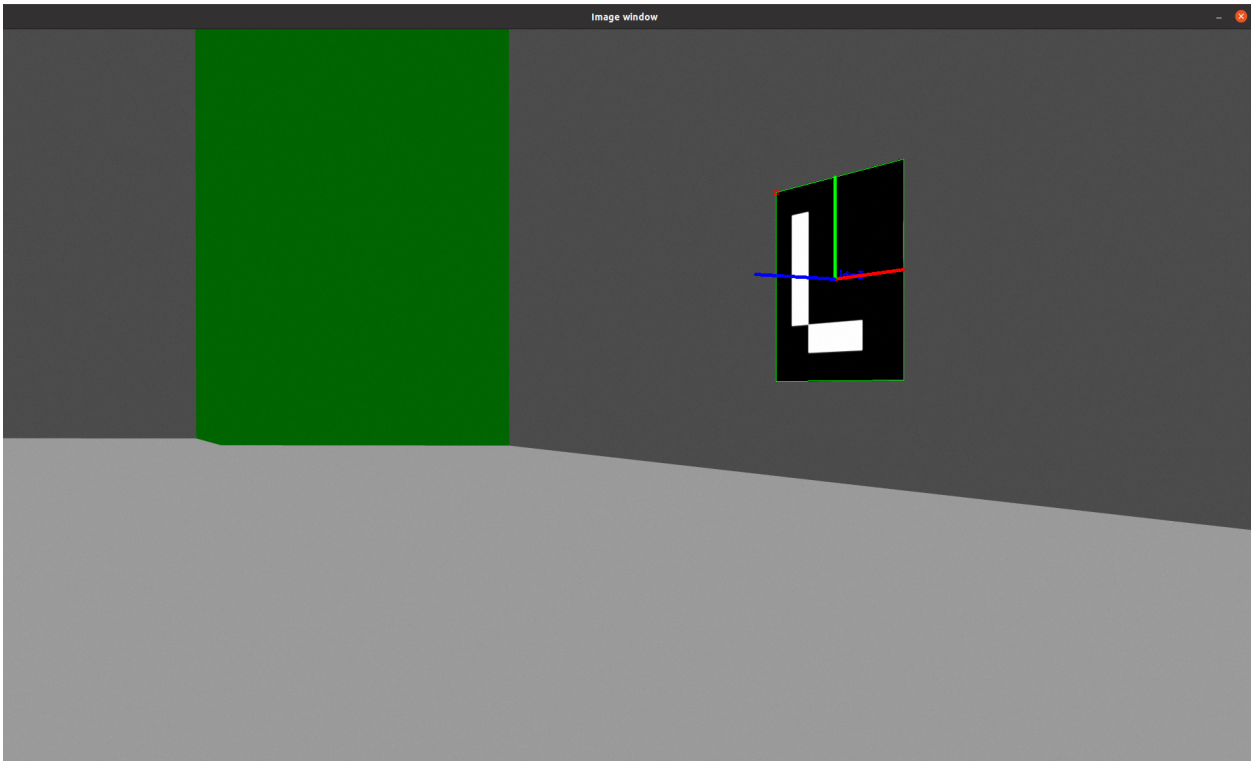

Figure 1: OpenCV Aruco Marker Detection

Figure 2 shows an overhead view of the robot in the simulation space.

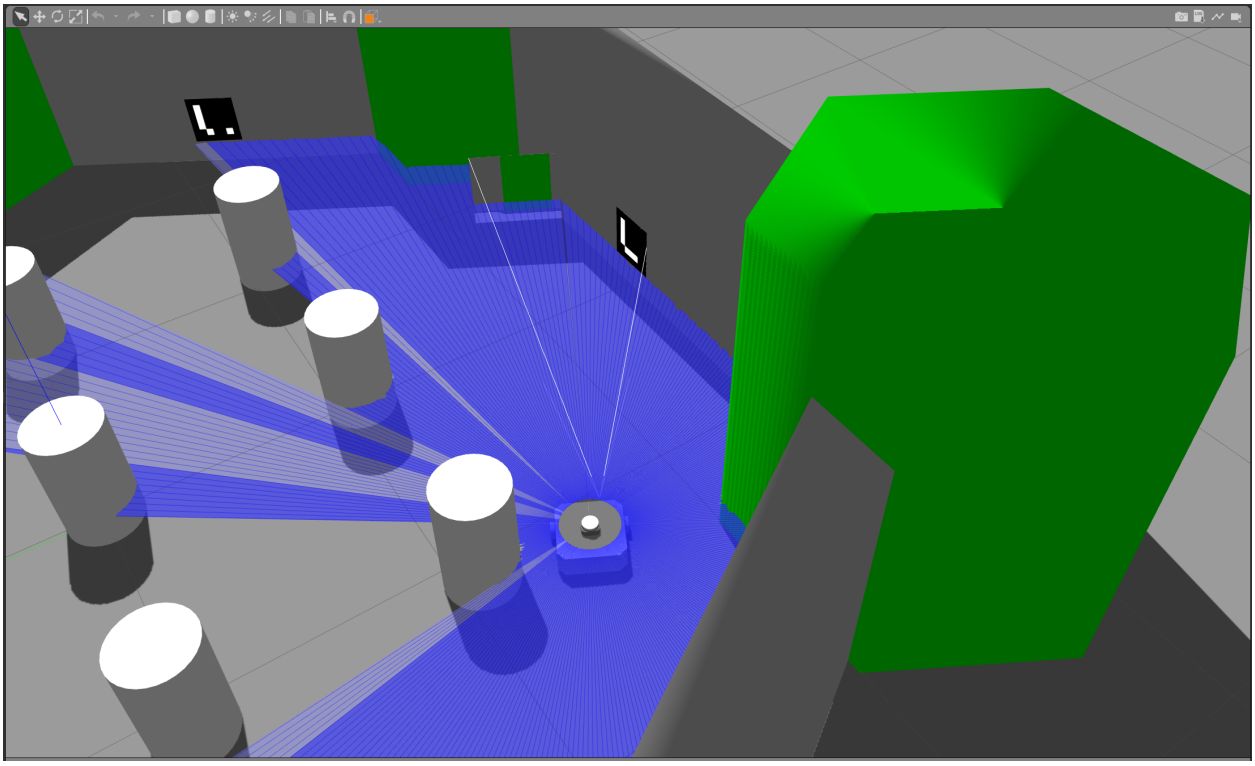

Figure 2: Bird's Eye View of the Simulation

The white angled lines coming out from the robot indicate the camera's field of view. In addition to the 3D coordinate transform, the OpenCV library provides the ID number of the detected aruco marker. Each of the placed aruco markers in the simulation space are unique, there are no duplicates. Since I placed these aruco markers myself, I know their exact coordinate transform in the world frame. Due to the nature of homogenous transforms in that you can chain them together, I can multiply the 4x4 homogenous transform from the world frame origin to the aruco marker by the homogenous transform of the aruco marker to the camera lens and in turn the transforf of lens to the base foot print of the robot. This means that when I detect an aruco marker in my camera frame I can calculate the location of the robot in the world frame coordinate system. The X and Y coordinates of both the detected aruco marker and the calculated X and Y coordinates of the robot serve as the basis of my observation. My observation is the two argument arc-tangent, commonly referred to as "atan2" using the robot and the aruco marker's coordinates. Equation 1 shows the equation for this observation.

$$\sigma = atan2((Y_R - Y_A), (X_R - X_A)) \tag{1}$$

Where (X_R, Y_R) are the X and Y coordinates of the robot, and (X_A, Y_A) are the X and Y coordinates of the aruco marker. Figure 3 shows a visual representation of the observation.

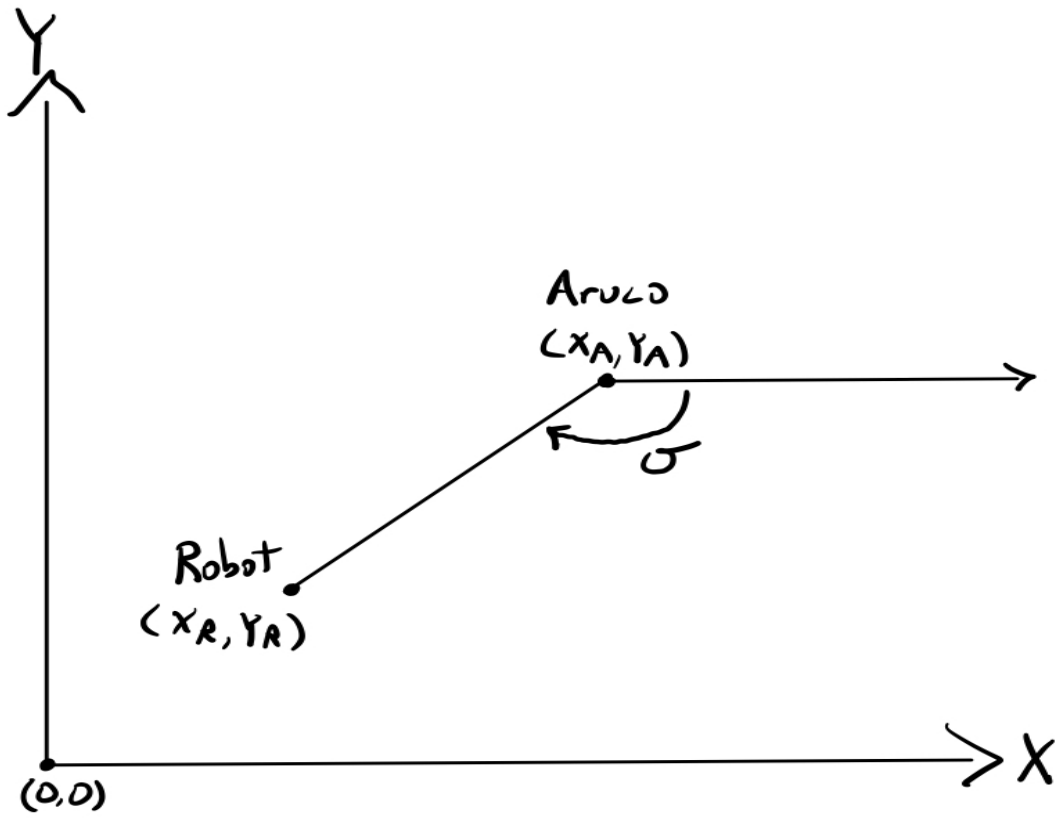

Figure 3: Sigma Observation Depiction

The Results:
The Extended Kalman Filter was implemented in real time using the Robot Operating System to pull in observed sigma angle calculations alongside the observed location of the aruco marker and robot using the methods described in the previous section. The output estimated state of the EKF was recorded using MATLAB and the results were compared to the ground truth values of the position and pose coming from the simulation. The results of are shown below in figures 4, 5 and 6. The infinite slope line in the yaw angle measurement is due to the fact that the simulator wraps it's yaw angle from $-\pi$ to π. I implemented this in my code to match.

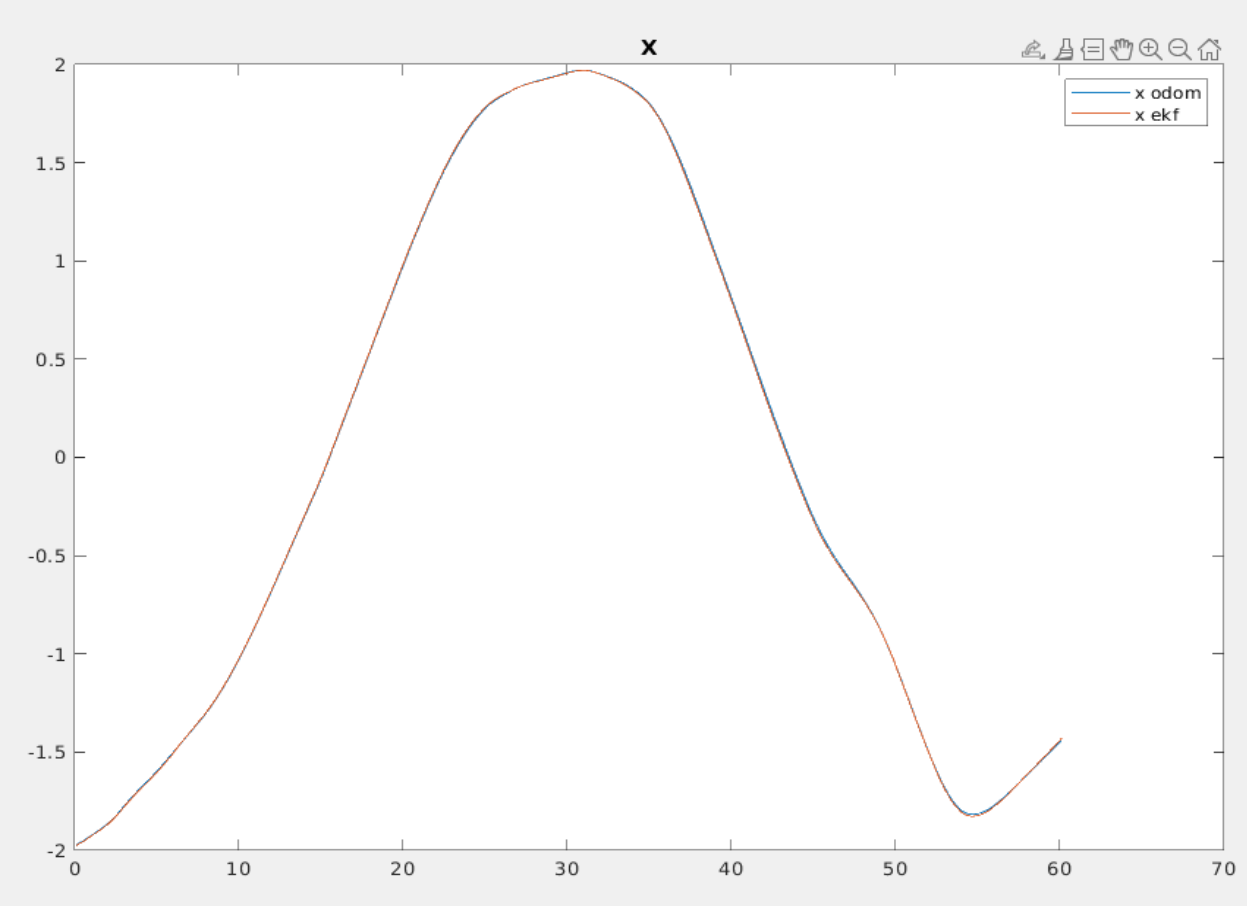

Figure 4: X Coordinate EKF Estimation vs. Ground Truth

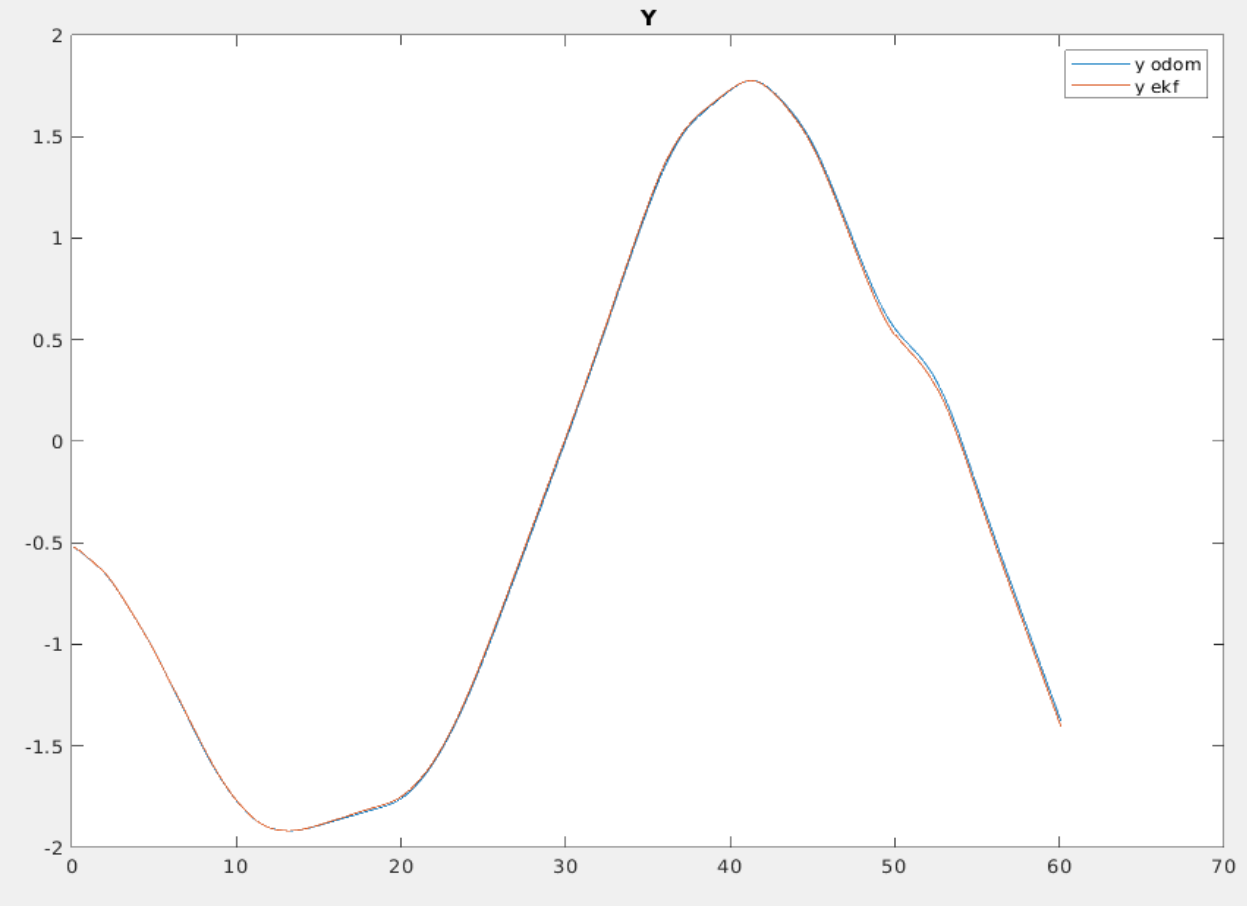

Figure 5: Y Coordinate EKF Estimation vs. Ground Truth

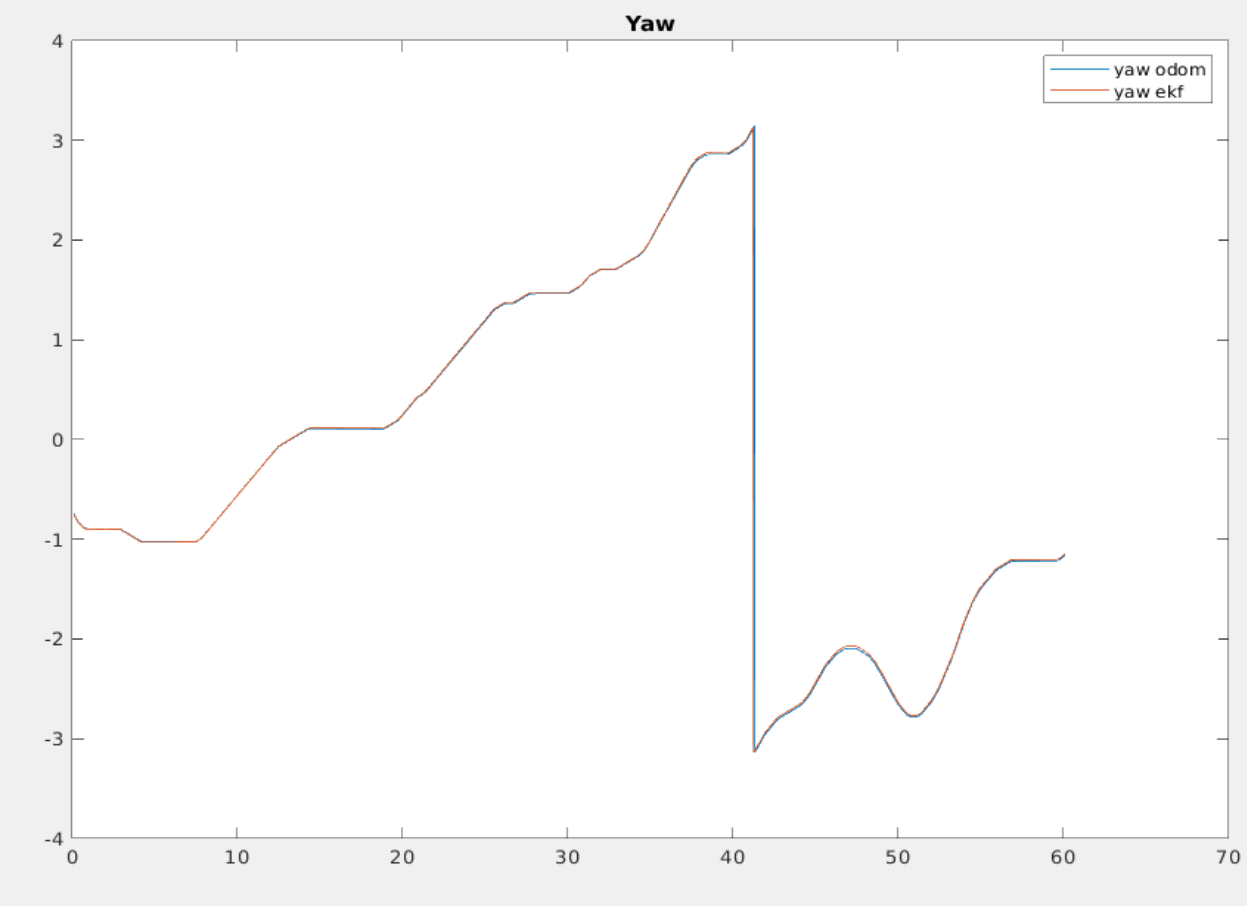

Figure 6: Yaw Coordinate EKF Estimation vs. Ground Truth

The results are honestly significantly better than I expected. I believe being able to run the code at a very high frequency of 30 times per second is doing a lot of work in reducing the error between the ground truth and the estimated state of the robot.