Control of Franka Emika Panda Robotic Arm Using Arduino IMU

David Yackzan
ME 699 Robotic Modeling and Control
University of Kentucky
Lexington, KY USA
dwya222@uky.edu

***Abstract*—An Arduino Nano was connected to a BNO055 9 Degree of Freedom (DOF) Inertial Measurement Unit (IMU) for inertial data measurement. The data was received and filtered on the Arduino Nano using a Kalman filter in Arduino/C++ code. The filter determined a roll, pitch, and yaw calculation for the current state of the IMU and these measurements were published to the serial port. Two Julia programs were written to read the measurements published to the serial port and simulate movement of a Franka Emika Panda Robot based on the calculated/measured angles of roll, pitch, and yaw from the Arduino. The end effector position was incremented based on the values and the inverse kinematics problem was solved each time the values received were above a certain threshold.**

***Index Terms*—IMU, inverse kinematics, control, Kalman filter, simulation, panda robot**

I. Introduction

Control of 6 and 7 DOF robotic arms through the use of teach pendants and standard controllers is often a tedious and non-intuitive task. These typical methods of control like may require a higher level of technical expertise than many people who would like to make use of a robotic arm have. And use of a standard controller may not require much expertise, but may require a user to undergo extensive amounts of training before they are able to perform precision tasks consistently. For these reasons novel and intuitive control methods for robotic arms are important.

Robotic arms have many repetitive applications in environments such as manufacturing assembly lines for fabrication and shipping warehouses for sorting, moving, and packaging. These types of tasks call for more automation and are less reliant on human proficiency with control methods. However, when robotic arms are needed for tasks that require human intervention such as in surgical robotics, explosive disarming, or radioactive material disposal, novel and intuitive control methods hold great potential for improving functionality.

Several systems have been developed that incorporate inertial control schemes attached to a human user's arm in order to actuate novel humanoid robotic arms as in [1] [2] [3]. The majority of work concerning inertial control of robotic arms has consisted of projects that implement forward kinematic inertial controls where the orientation signals from the IMU are directly mapped to joint angles on the robotic arms. While this may be useful for some applications, such as manipulation of a lower-DOF robotic arm inside of a constrained space where each joint position must able to be individually controlled, it is a less intuitive approach than applying inverse kinematic controls. The use of inverse kinematic controls allows for direct control of the end effector of a robot, which allows for the user to focus more fully on the task space rather than attempting to manually map joint motion to end effector motion. In order to exploit these advantages an inverse kinematic control scheme was developed for this project.

Another novel feature of this experiment is in its application to the Franka Emika Panda robotic arm. The current body of research on inertial control of robotic arms largely makes use of humanoid, 3-DOF robotic arms as in [1] or other simplified, low-DOF robotic mechanisms as in [2]. In situations where inertial control schemes have been applied to higher DOF arms as in [3], it has not been applied with inverse kinematic control.

The Panda robot was chosen for this project due to its reputation for being state-of-the-art. It is known for its protean nature and ability to complete various tasks and achieve various orientations within its work space, so developing a novel control scheme for such a mechanism only stands to add versatility to versatility. Outlining a strategy to translate the motion of a human arm to that of a fully developed robotic arm will also be useful for the development of future research concerning both the Panda robot as well as other complex robotic arms like it.

II. System Overview

The hardware design of this system consists of an Arduino Nano, a 9-DOF BNO055 IMU sensor, a breadboard, a few short wires, and a USB to Mini USB cable to connect the system to a computer. The wiring is shown in the schematic in Figure 1 and the image in Figure 2 on Page 2.

The software versions used for this system were the Arduino IDE version 1.8.14 and Julia version 1.5.3. Three programs were developed to achieve the goals of the experiment. One was written in C++ in the Arduino IDE and was uploaded to the Arduino Nano that accessed the IMU data, filtered it, and printed it to the serial port as three float numbers corresponding to the roll, pitch, and yaw of the sensor. The second was written in Julia that displayed a Franka Emika Panda robotic arm model, and defined a function with the ability to solve the inverse kinematics problem. The third program was also written in Julia that received the most recent

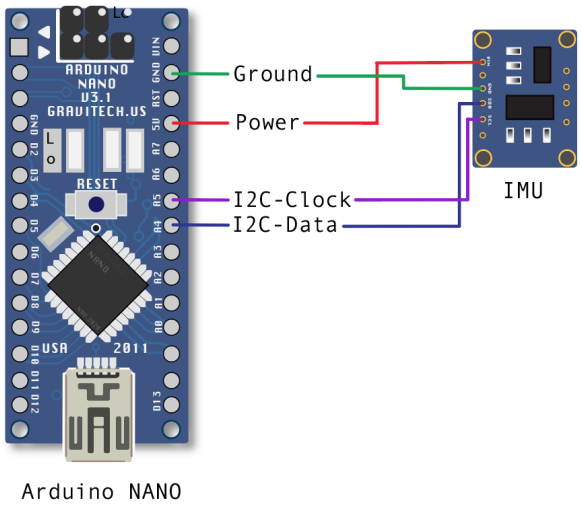

Fig. 1. Wiring Schematic

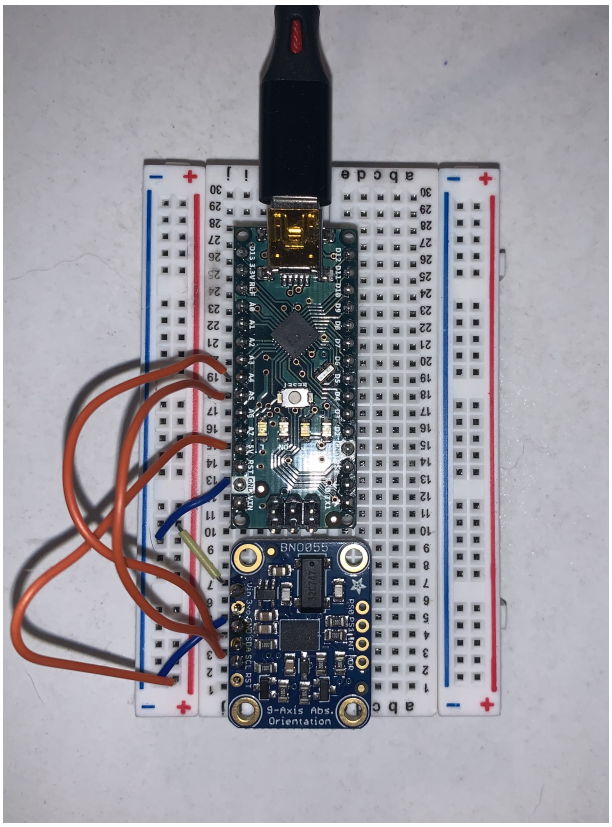

Fig. 2. Circuit Image

data from the serial port, interpreted it to determine the desired end effector motion, and simulated motion of the arm by using the inverse kinematics solution outlined in the first program.

III. Methods

Two different filters were applied on the Arduino with the data retrieved from the IMU. A simple complementary filter and a more complex Kalman Filter. Details on the implementation of both are outlined below.

A. The Complementary Filter

The complementary filter is a simple filter that balances the propensity of gyroscope sensors to be impacted by drift and the propensity of accelerometer sensors to be impacted by noise and vibration. It is comprised of a low-pass filter for the accelerometer measurements combined with a high-pass filter for the gyroscope measurements. The low frequency accelerometer readings that factor into the final value balance the drift that is seen in the gyroscope readings. The equation for the complementary filter is as follows:

$$z_k = k_{gyro} * (z_{k-1} + z_{gyro} * \Delta t) + k_{acc} * z_{acc}. \quad (1)$$

Where z_k is the filtered measurement at the current time k, k_{gyro} is the gain placed on the sum of the previous reading with the new gyrometer reading, $z_{gyro} * \Delta t$, and k_{acc} is the accelerometer gain placed on the accelerometer reading z_{acc}. The gains used in this experiment were $k_{gyro} = 0.99$ and $k_{acc} = 0.01$.

B. The Kalman Filter

The state equations used for the IMU are as follows:

The State Equation:

$$x_k = Ax_{k-1} + Bu_k + w_k \quad (2)$$

Where x_k is the state of the system at time k, x_{x-1} is the state of the system at the previous time step $k-1$ which is comprised of the previous angle θ and the gyroscope bias $\dot{\theta}_b$ given by

$$x_{k-1} = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}.$$

A is the state transition model given by

$$A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}.$$

B is the control-input model given by

$$B = \begin{bmatrix} dt \\ 0 \end{bmatrix}.$$

u_k is the control input at time k which is the gyroscope measurement in degrees per second given by

$$u_k = \dot{\theta}.$$

And w_k is process noise characterized by a Gaussian white noise (GWN) distribution with a covariance Q_k:

$$w_k \sim N(0, Q_k)$$

where

$$Q_k = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}} \end{bmatrix}.$$

The system was found to be responsive to change and robust to noise at Q_θ and $Q_{\dot{\theta}}$ values of 0.05^2 and 0.0001^2 respectively.

The Measurement Equation:

$$z_k = Hx_k + v_k \tag{3}$$

Where z_k is the measurement of the state x_k determined from the sensor values. H, the observation model, is used to map the state space x_k to the observed space and is given by

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

And w_k is process noise characterized by a GWN distribution with a covariance $R = .03$:

$$v_k \sim N(0, R).$$

Reference [4] was used to determine the Kalman Filter equations which are as follows:

The State Prediction Equation:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + B\dot{\theta}_k \tag{4}$$

Where $\hat{x}_{k|k-1}$ is the prediction of the current state based on the previous state and $\dot{\theta}_k$ is the measurement from the gyroscope at time k in degrees per second.

The Covariance Prediction Equation:

$$P_{k|k-1} = AP_{k-1}A^T + Q_k \tag{5}$$

Where $P_{k|k-1}$ is the estimate of the error covariance of the current state based on the covariance of the previous state P_{k-1}.

The Innovation Equation:

$$\tilde{y}_k = z_k - H\hat{x}_{k|k-1} \tag{6}$$

Where $\tilde{y}_k$ represents the difference between the measurement z_k and the estimated state $\hat{x}_{k|k-1}$. For this application, the z_k used was the same z_k that comes from the complementary filter in equation 1.

The Innovation Covariance Equation:

$$S_k = HP_{k|k-1}H^T + R \tag{7}$$

Where S_k is an estimation of how trustworthy the measurement is based on $P_{k|k-1}$.

The Kalman Gain Equation:

$$K_k = P_{k|k-1}H^T S_k^{-1} \tag{8}$$

Where the Kalman gain at the current state K_k is the weight put on the measurement of the state vs. the previous state based on the error covariance $P_{k|k-1}$ and the innovation covariance S_k.

The Posteriori Estimate Equation:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_k \tag{9}$$

Where $\hat{x}_{k|k}$ is the posteriori estimate of the current state determined from the combined sum of the proiri state $\hat{x}_{k|k-1}$ and the innovation $\tilde{y}_k$.

The Posteriori Error Equation:

$$P_{k|k} = (I - K_kH)P_{k|k-1} \tag{10}$$

Where $P_{k|k}$ is the posterior error and I is the identity matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

C. Implementation

The above Kalman gain equations were implemented into the Arduino code as functions for both the pitch and the roll state estimations of the IMU. Each time a new measurement was received from the IMU sensor, the complementary filter was applied to filter out noise and drift from the readings and then the Kalman functions were called using the filtered pitch and roll to find an accurate estimate of the actual state of the IMU. These values were printed to the serial port where the Julia program was able to read them.

The startup Julia program was run to pre-compile the necessary packages and initialize the visualization of the Panda robotic arm model as seen in Figure 3 on Page 4. Once initialized, the main Julia program was run that looped through the process of reading and assigning the roll, pitch, and yaw values from the serial port to variables, determining whether the values were large enough to require a change in the end effector position. If the values were above the threshold, the program implemented the corresponding increment or decerement to the end effector position via the solution to the inverse kinematics problem and visualized the motion.

The threshold for pitch and roll was 45 degrees from the zero state, or level state. So if the IMU was pitched or rolled more than +45 degrees or less than -45 degrees, then the threshold was exceeded and the desired position of the end effector was incremented or decremented. As for the magnetometer, since the user is not expected to know where 0 degrees (or due South) is, the initial orientation of the magnetometer was measured and the threshold was set as 45 degrees plus or minus that initial position. The pitch motion of the IMU was mapped to the x-axis end effector position (forward and backward) of the Panda model, the roll motion of the IMU was mapped to the z-axis end effector position (up and down), and the yaw motion of the IMU was mapped to the y-axis end effector position (side to side).

IV. Results

The state estimates were found to be extremely accurate and responsive to the actual state of the IMU using the filter method outlined in the methods section. The Kalman

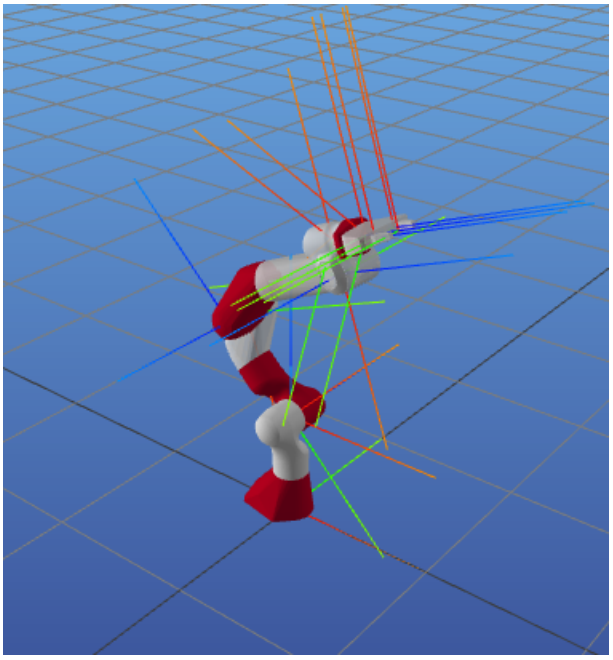

Fig. 3. Franka Emika Panda Robotic Arm Model

filter was able to follow the state of the IMU responsively on its own as well, but was not as robust to vibration and noise without the added implementation of the complementary filter. This can be seen in the comparison between the two figures below, where Figure 4 shows the response of the system to perturbation without the complementary filter and Figure 5 shows the response of the system to perturbation with the complementary filter. While the system without the incorporation of the complementary filter fluctuates about 30 degrees of assumed tilt, the system with the complementary filter only fluctuates a few degrees.

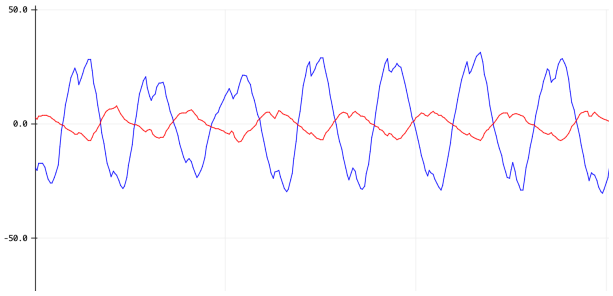

Fig. 4. Without Complementary Filter

Due to the accuracy and responsiveness of the combined filtering methods, the finished system was both responsive and intuitive. The end effector position of the Panda robotic arm model was able to be controlled with the IMU controller to move to any position within the task space of the model quickly and easily. The motion could be stopped at any time by returning the IMU to level at its relative starting orientation.

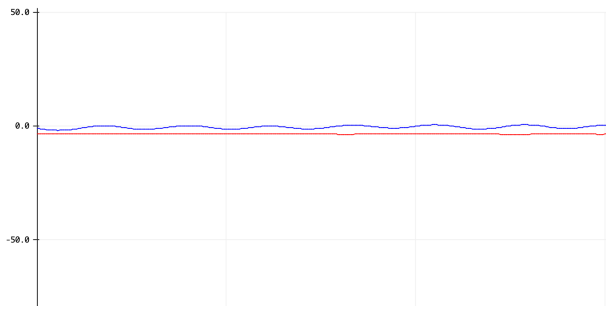

Fig. 5. With Complementary Filter

V. Conclusion And Future Work

The ease of use of the final system shows that there is potential for this type of control scheme to be used in situations where novel and intuitive controllers could be beneficial, however more research must be conducted before this design is ready to be used to control a physical robotic arm. The robotic arm model in this experiment was rendered as massless and therefore joint angles could be manipulated to desired values without inertial or force considerations. For implementation on a real robotic arm, mass of joints must be taken into account, and a control scheme for actuation, such as a proportional derivative (PD) controller, must be designed. This system also only included methods to control the end effector position, and not the orientation. Future research could include the implementation of a secondary IMU that maps its orientation to the end effector orientation to allow for full end-effector control.

If these two pieces of future work are added then there is very good potential for this design to be implemented usefully anywhere that non-automated control of panda robotic arms is useful. This could be for applications like the disarming of bombs, hazardous material disposal, or even surgical robotics. This design of inertial control proved to be responsive, intuitive, and accurate which are paramount factors in determining the viability of control schemes.

Acknowledgment

I would like to thank my advisor Dr. Hasan Poonawala and my lab mate Benton Clark for their assistance in figuring out how to use the amazing Julia language to access serial port information line by line (by using Python).

References

[1] R. Sekhar, R. K. Musalay, Y. Krishnamurthy, and B. Shreenivas, "Inertial sensor based wireless control of a robotic arm," in *2012 IEEE International Conference on Emerging Signal Processing Applications*, 2012, pp. 87–90.

[2] J. Jiang, A. McCoy, E. Lee, and L. Tan, "Development of a motion controlled robotic arm," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, 2017, pp. 101–105.

[3] P. Neto, J. N. Pires, and A. P. Moreira, "Accelerometer-based control of an industrial robotic arm," in *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, 2009, pp. 1192–1197.

[4] Lauszus. A practical approach to kalman filter and how to implement it. [Online]. Available: http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/