Estimation of Joint Velocity and Acceleration of a Robotic Manipulator Using Kalman Filtering and Discrete-Time Adaptive Windowing

Sahand Rezaei-Shoshtari, *Student, McGill University*

I. Introduction

ROBOTIC arms have received considerable attention in the recent years for their wide range of applications in the industry. As a result, a growing number of studies are now trying to achieve fully autonomous manipulation. One of the fields which has been relatively overlooked by researchers is autonomous forestry. This is probably due to the high complexity and nonlinearity of the environment.

This project is the starting point of a research to develop hybrid dynamic model of a cutting manipulator. The term hybrid means the model combines machine learning and data-driven modeling with classical Lagrangian dynamics. In other words, it tries to use the human knowledge as a priori to the learning method.

The robotic manipulator used in this research is Kinova Jaco^2 6-DOF arm. This lightweight arm was originally designed as an assistive robot, thus, it fits in general grasping and pushing applications.

Fig. 1. Jaco^2 6-DOF robotic arm, designed and manufactured by Kinova Robotics [1]

For the data-driven approach, joint velocity and acceleration are required. However, the joint encoders are only able to provide joint position, therefore, velocity and acceleration should be estimated. There are three main approaches in the literature: finite difference methods, filtering methods, and adaptive windowing. Finite difference methods perform poorly in high sampling rates; therefore, Kalman filtering and adaptive windowing have been implemented in this project, and their performance have been compared.

S. Rezaei-Shoshtari is with the Department of Mechanical Engineering, McGill University, Montreal, QC, H3A 0C3 Canada e-mail: sahand.rezaei-shoshtari@mail.mcgill.ca

The remainder of this paper is structured as follows. Section II provides a background on finite difference methods, section III describes the filtering approach and section IV presents the adaptive windowing methodology. Section V provides the details behind modeling and simulation, while section VI presents the estimation results.

II. Finite Difference Methods

Without loss of generality, we first consider the case of estimating velocity from position signal $\theta(t)$, since all of the discussed methods can be applied to estimate the acceleration from angular velocity signal $\omega(t)$.

Suppose $\theta(t)$ is sampled with period T and $y_k = \theta_k + e_k$ is the measurement, where e_k is a zero mean bounded error. In most of the cases this assumption is valid and thus we can safely present the error by $-d \leq e_k \leq d$. The problem at hand is to estimate $\hat{\omega}_k$ from measurements $\{y_i\}_{k-n}^{k}$, in which n is the size of the sampling window [2]. Since the estimate is to be used online, the estimation should be carried out in real-time. One of the approaches in the literature is Finite Difference methods (FDM) [3].

The Finite Difference method uses Euler approximation:

$$\hat{\omega}_k = \frac{y_k - y_{k-1}}{T} = \frac{\theta_k - \theta_{k-1}}{T} - \frac{e_k - e_{k-1}}{T} \tag{1}$$

Finite Difference methods become unstable at high sampling rates, because as T decreases, the error term increases, while the position term stays more or less the same. To further describe this problem, consider an encoder system with an interpulse angle θ_m and sampling rate T. If we suppose N_p is the number of pulses occurring during interval T, $N_p\theta_m/T$ would be the angular velocity estimate [4]. In the encoders of Jaco^2 arm with $\theta_m = 0.0002^\circ$ and 1-KHz sampling frequency, the resolution would be $0.2^\circ/sec$, which is not acceptable for precise application. This is even worse in acceleration estimate as the resolution would be $\theta_m/T^2 = 2000^\circ/s^2$.

The variance of velocity and acceleration estimates are found in [4] to be as follows:

$$var[\hat{\omega}_k] = \frac{2}{3}\frac{\theta_m^2 + 2r}{T^2} \tag{2}$$

$$var[\hat{\alpha}_k] = \frac{4}{3}\frac{\theta_m^2 + 2r}{T^4} \tag{3}$$

Where,

$$r = \mathbb{E}[e_k^2] = \frac{1}{3}d^2 \tag{4}$$

The variances increase drastically as T decreases, making the Finite Difference method highly unreliable.

III. Filtering Methods

As the Finite Difference method cannot provide accurate estimates, applying filtering is necessary in order to reduce the variance of the estimates. In the least-squares sense, the Kalman filter is the optimal filter.

A. The Motion Model

Since dynamics-based motion model requires information on the masses, inertia parameters, joint friction and actuator models, the kinematics-based approach is taken here. As most of the required information are either rough approximates or not provided by Kinova Robotics.

The motion model can be described using canonical state description [5]:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{\Gamma}\mathbf{w}(\mathbf{t}) \tag{5}$$

With,

$$\mathbf{A} = \begin{bmatrix} -a_{n-1} & 1 & 0 & \cdots & 0 \\ \vdots & & & & \\ -a_1 & 0 & 0 & \cdots & 1 \\ -a_0 & 0 & 0 & \cdots & 0 \end{bmatrix}; \mathbf{\Gamma} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

Where the state variables $\mathbf{x}(t) = [x_1 \ x_2 \ x_3]^\intercal$ are angle, angular velocity and angular acceleration, respectively. $\mathbf{w}(t) = [w_1 \ w_2 \ w_3]^\intercal$ is a zero-mean white noise and $\mathbf{\Gamma}$ is defined as above.

B. The Measurement Model

The measurement model can be presented by the following equation [5]:

$$y(t) = \mathbf{C}\mathbf{x}(t) + e(t) \tag{6}$$

With,

$$\mathbf{C}^\intercal = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$$

Where the scalar output $y(t)$ is the measurement, that is the latest encoder count multiplied by the resolution θ_m. The error $e(t)$ is the quantization error, represented by a bounded zero-mean white noise with variance r from equation (4).

C. The Asymptotic Analysis

Using discrete-time Riccati equation, it can be proved that at high sampling rates, the discrete Kalman filter is independent of $a_0, a_1, \cdots, a_{n-1}$. As a result, the a_i might as well set to be zero, making the model a chain of integrators. The proof can be found in [5] and is out of the scope of this paper.

D. Discretization

Based on the results obtained from the asymptotic analysis and using Ricatti equation, an n-integrator discretized model can be described by (7).

$$\mathbf{A}_d = \begin{bmatrix} 1 & T & \frac{T^2}{2!} & \frac{T^3}{3!} & \cdots & \frac{T^{n-1}}{(n-1)!} \\ 0 & 1 & T & \cdots & \cdots & \frac{T^{n-2}}{(n-2)!} \\ \vdots & & & & & \\ 0 & 0 & \cdots & \cdots & 1 & T \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \tag{7}$$

On the other hand, the model presented by 5 and 6 can be discretized as in [2]:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{L}\mathbf{w}_{k-1} \tag{8}$$

$$y_k = \mathbf{C}\mathbf{x}_k + e_k \tag{9}$$

Where same as before, the state variables $\mathbf{x}(t) = [x_1 \ x_2 \ x_3]^\intercal$ are angle, angular velocity and angular acceleration. $\mathbf{A}$ and $\mathbf{C}$ are the state transition and observation matrices, respectively. Since both acceleration and velocity are to be estimated, a triple integrator needs to be used.

$$\mathbf{A} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$\mathbf{w}_k = [w_{1k} \ w_{2k} \ w_{3k}]^\intercal$ is the process noise and and e_k is the measurement noise. $\mathbf{w}_k$ can be viewed as a surrogate for the derivative of acceleration in the triple integrator case. $\mathbf{L}$ is usually defined as the identity matrix. The noises are assumed to be zero-mean white Gaussian. The covariance of $\mathbf{w}_k$ is defined as $\mathbf{Q}_k\delta_{kj} = \mathbb{E}[\mathbf{w}_k\mathbf{w}_k^\intercal]$. In the triple integrator case, since $\mathbf{Q}_k$ is regarded as a surrogate for the derivative of acceleration, it can be rewritten as $\mathbf{Q}_k = diag[0 \ 0 \ q]$. Parameter q should be considered as a parameter to be adjusted, because actual motions are not well represented by stationary random process. Same as before, the covariance of the measurement noise is defined as r from equation (4) and is a scalar [2].

E. Observability

In order for the system to be observable, matrix $\mathbf{Q}_0$ must be full-rank.

$$\mathbf{Q}_0 = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA^2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & T & \frac{T^2}{2} \\ 1 & 2T & 2T^2 \end{bmatrix}$$

$$\det\mathbf{Q}_0 = T^3 \neq 0 \Rightarrow \text{rk}\mathbf{Q}_0 = 3$$

Thus, the system is observable.

F. Kalman Filter

The discrete-time Kalman filter can now be applied to the system described by equations (8) and (9).

Prediction:

$$\check{\mathbf{x}}_k = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1} \tag{10}$$
$$\check{\mathbf{P}}_k = \mathbf{A}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{A}_{k-1}^\intercal + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^\intercal$$

Gain:

$$\mathbf{V}_k = \mathbf{C}_k\check{\mathbf{P}}_{k-1}\mathbf{C}_{k-1}^\intercal + r$$
$$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{C}_k^\intercal\mathbf{V}_k^{-1}$$

Correction:

$$\check{y}_k = \mathbf{C}_k\check{\mathbf{x}}_k \tag{11}$$
$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(y_k - \check{y}_k)$$
$$\hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{C}_k)\check{\mathbf{P}}_k$$

Note that $\mathbf{M}_k$ is the identity matrix in the system model and thus does not show up in the Kalman filter equations.

IV. Adaptive Windowing Methods

The Euler approximation on two samples is more **precise** if they are far apart, since larger window length results in smaller variance. This is similar to averaging the last n velocity estimates $\hat{\omega}_k, \hat{\omega}_{k-1}, \cdots, \hat{\omega}_{k-n}$ with $\hat{\omega}_i$ obtained from finite difference method [2],

$$\hat{\omega}_k = \frac{1}{n}\sum_{j=0}^{n-1}\hat{\omega}_{k-j} = \frac{y_k - y_{k-n}}{nT} \tag{12}$$

On the other hand, larger window size is analogous to decreasing the sampling rate. Consequently, it introduces time delay and reduces the estimation **reliability**. Thus, the window size should be selected adaptively in order to balance the trade off between reliability and precision. The window size should be set to small when the velocity is high in order to achieve reliable and fast estimates. Additionally, the window size should be large when the velocity is low so that the estimates are precise [2].

In other words, precision puts a lower bound on the window size, whereas reliability presents an upper bound on it. We have to describe a criterion based on which the longest window size is found while the reliability is maintained at a satisfactory level. The criterion may be as simple as ensuring a straight line passing through y_k, y_{k-n} covers all intermediate samples with an uncertainty band defined by the maximum of the noise $d = ||e_k||_\infty$. [2].

A. End-Fit First-Order Adaptive Windowing

The first proposed method by [2] is "end-fit first-order adaptive windowing" (end-fit-FOAW). In this method, the problem is described as finding a window of length n where $n = \texttt{max}\{1, 2, 3, ...\}$ such that

$$|y_{k-i} -^L y_{k-i}| \leq d, \forall i \in \{1, 2, 3, ...\} \tag{13}$$

Where ${}^L y_{k-i} = a_n + b_n(k-i)T$ with

$$a_n = \frac{ky_{k-n} + (n-k)y_k}{n} \tag{14}$$
$$b_n = \hat{\omega}_k = \frac{y_k - y_{k-n}}{nT} \tag{15}$$

The optimality of the method is proved in [2] by the following preposition.

Preposition: If a position trajectory has a piecewise continuous and bounded derivative, and if the measurement noise is uniformly distributed, the proposed method minimizes the velocity error variance and maximizes the accuracy of the estimate.

Proof: Let χ be a position trajectory with a piecewise continuous and bounded derivative V. The time derivative at the instant k is denoted by ω_k. Since the measurement noise is uniformly distributed, (12) yields:

$$\hat{\omega}_k \in [\frac{y_k - y_{k-n}}{nT} - \frac{2d}{nT}, \frac{y_k - y_{k-n}}{nT} + \frac{2d}{nT}], \texttt{or}$$
$$\hat{\omega}_k - \frac{2d}{nT} \leq \omega_k \leq \hat{\omega}_k + \frac{2d}{nT} \tag{16}$$

Which in turn results in the probability density function (PDF) presented in Figure 2. Variance of angular velocity error is thus presented by $\mathbb{E}[w_k^2] = \frac{2d^2}{3n^2T^2}$.

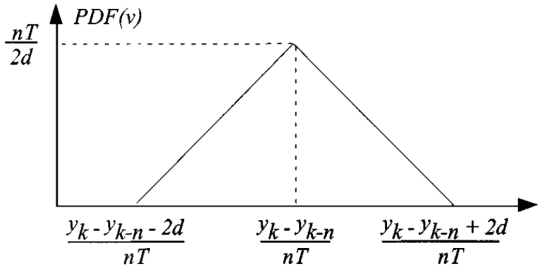

Fig. 2. PDF of angular velocity for window length n [2].

Hence, as n increases, the variance for angular velocity error decreases. In addition, the slope of the line passing through the points y_k and y_{k-n} is the estimate of the maximum likelihood [2].

The pseudo code for end-fit-FOAW algorithm is then as follows:

Algorithm 1 end-fit-FOAW

```
function VELOCITY ESTIMATOR
top:
    Set i = 1.
loop:
    Set y_k as the last sample and y_{k-i} as the ith before y_k.
    Calculate b_n, slope of the line passing through y_k and y_{k-i} from (15).
    if The line passes through all points inside the window within the uncertainty band of each point then
        Set i ← i + 1
        goto loop
    else
        Return the last estimate.
```

B. Best-Fit First-Order Adaptive Windowing

The end-fit-FOAW works by estimating the velocity using two position measurements. Therefore, when the window size

is small, it may overshoot. The authors in [2] have also proposed another method called "best-fit-FOAW" in which the velocity estimate is the slope of a least square approximation. This estimation minimizes the error energy, and thus is a natural choice. The best-fit-FOAW is the same as end-fit-FOAW but differs in *line 6*, where b_n is calculated. Instead the following formula is proposed for the calculation of b_n [2]:

$$b_n = \frac{n\sum_{i=0}^{n} y_{k-i} - 2\sum_{i=0}^{n} iy_{k-i}}{Tn(n+1)(n+2)/6} \tag{17}$$

Similar to end-fit-FOAW, the optimality of the best-fit-FOAW can be proved.

V. Simulation

As the Kinova Jaco^2 arm [1] is ROS (Robot Operating System) compatible, the simulation and dynamics modeling is carried out in ROS/Gazebo framework. ROS creates an environment for developing robot applications by providing hardware abstraction, device drivers, libraries, visualizers, message-passing, and package management [6]. On the other hand, Gazebo is a 3D dynamic simulator with the ability of accurately and efficiently simulating populations of robots in complex indoor and outdoor environments [7]. The perfect integration of Gazebo into ROS enables them to be the most powerful robotic software.

The Universal Robotic Description Format (URDF) is an XML file format used in ROS to describe all elements of a robot. The URDF of Jaco^2 arm is provided by Kinova Robotics. The model contains the inertia parameters of the links and actuators. Mass and center of mass (COM) of each link are very accurate, however the inertia parameters are calculated by assuming uniform cylinders for each link. In addition, the joint dynamics parameters, such as damping, friction and stiffness, do not accurately represent the hardware. As a result, the simulation might have some deviations from the reality, however, for the purpose of this research, the results are accurate enough.

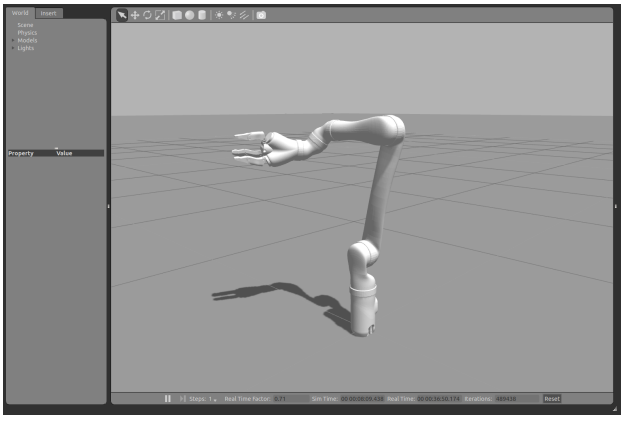

Fig. 3. Simulation of the Jaco^2 arm in Gazebo Software.

One of the powerful features of ROS and Gazebo is their ability to simulate sensors. Gazebo provides models of many common sensors. By default, it models each sensor (except the IMU sensor) perfectly. However, it is capable of adding noise to the measurement system in order to make it more realistic [7].

Another software which is used in this research is MoveIt! [8]. MoveIt! runs on top of ROS and provides functionality for kinematics, motion/path planning, collision checking, 3D perception, and robot interaction. Furthermore, RViz is a powerful visualization tool which enables the programmer to see the world through the eyes of the robot. This ROS package communicates with the ROS master and gathers sensory data and camera images from the robot; it then visualizes all the measurements through a user-friendly GUI. Additionally, it provides users with several other useful tools, such as keeping track of the frames of each link.

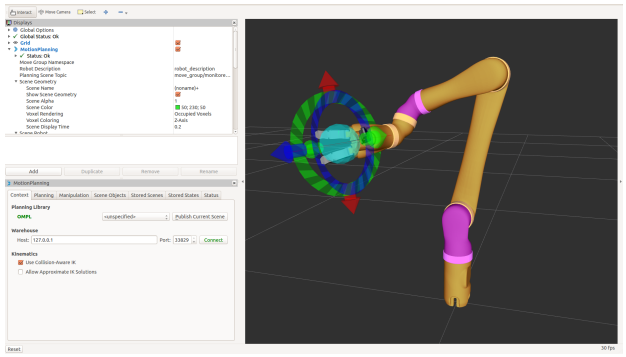

Fig. 4. Visualization of the Jaco^2 arm in RViz software. The user can command the robot by setting the position of the end-effector.

With all the available tools, both the dynamics simulation and sensor measurement are as close as possible to the real world. Therefore, if the angular velocity and acceleration can be carried out successfully in this framework, it will surely work on the real robot as well.

A. ROS Architecture

Before going into details of the implementation, some of the key concepts of ROS need to be described. The *Computation Graph* is a peer-to-peer network of ROS processes. The basic concepts of Computation Graph in ROS are as follows [6]:

- **Nodes:** Nodes are processes that perform computation. Each node is capable of working alone, ensuring the modular nature of a robotic system (i.e. sensors, actuators, controls, path planning, etc.).
- **Master:** The ROS Master provides name registration and look up to everything in the Computation Graph.
- **Messages:** Nodes communicate with each other via messages.
- **Topics:** A node sends out a message by simply *publishing* it in to a topic. Furthermore, other nodes can receive the message by *subscribing* to the desired topic. One can think of a topic as a message bus.
- **Services:** The *publish/subscribe* model is not suitable for *request/reply* interactions. The service providing node offers a service under a name and a client uses the service by sending the request message and awaiting the result.

An estimation task is based on the concept of *publish/subscribe* interactions. An estimator node subscribes to the

topic containing robot states, performs the estimation task and publishes the estimated velocity and acceleration to another topic. An important point regarding ROS messages is the possibility of data loss in the communications. In other words, a ROS message might not get to the subscriber due to network errors. As a result, the implementation must be robust so that it checks the time stamp and header of each message to check whether it has lost some of the data along the way.

VI. Results

In this section, the performance of the Kalman filter is compared with end-fit-FOAW and best-fit-FOAW. The scenario which was chosen is a simple pick and placement task. In this scenario the robot starts from the home position, moves towards the cylinder-shaped object, grabs and and takes it to another location and puts it down.

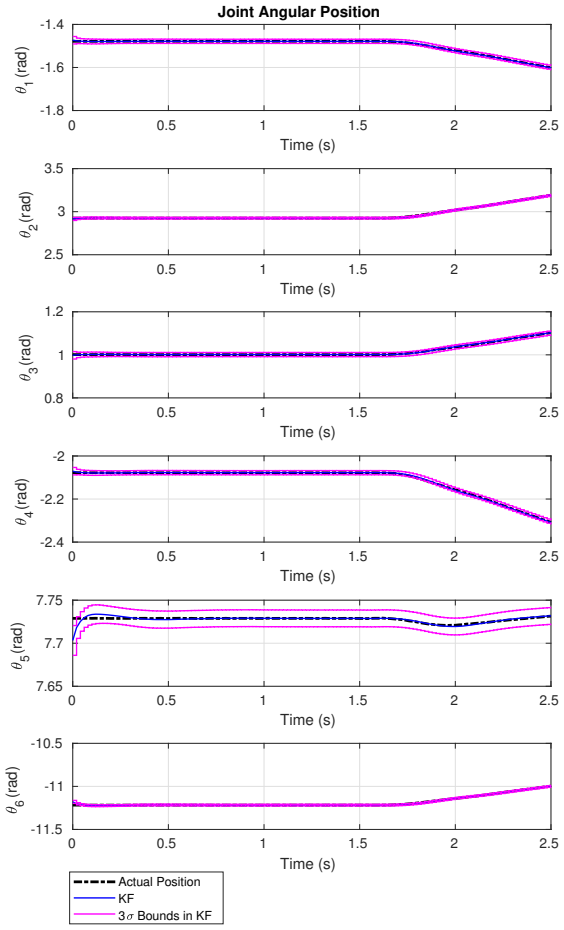

Fig. 5. Plots of the actual and estimated angular position using Kalman filtering during the first 2.5 seconds of the scenario. The 3σ bounds for the estimates are also provided.

A. Parameters

The angular position is measured at the rate of 50 Hz. Based on the specification sheets of the actuators used in Jaco^2 arm, the absolute position sensor precision at start-up is $\pm 1.5^\circ \approx 0.026$ radians, thus, a uniformly distributed error of magnitude 0.026 radians is added to the position measurements. The covariance of the measurement noise is given by (4). As discussed earlier, q in the Q matrix is an adjustment parameter working as a surrogate for acceleration and its derivative. The value of q has a huge impact on the performance of the Kalman filter; small values decrease the overshoot and increase the delay of the filter, whereas large values provide a much faster response time but with a downside of an overshoot. In addition, larger values for q cause a larger 3σ bounds, especially for the acceleration estimates. With multiple estimations, $q = 0.01$ was found to be the optimum q which keeps the balance between the overshoot and delay of the filter.

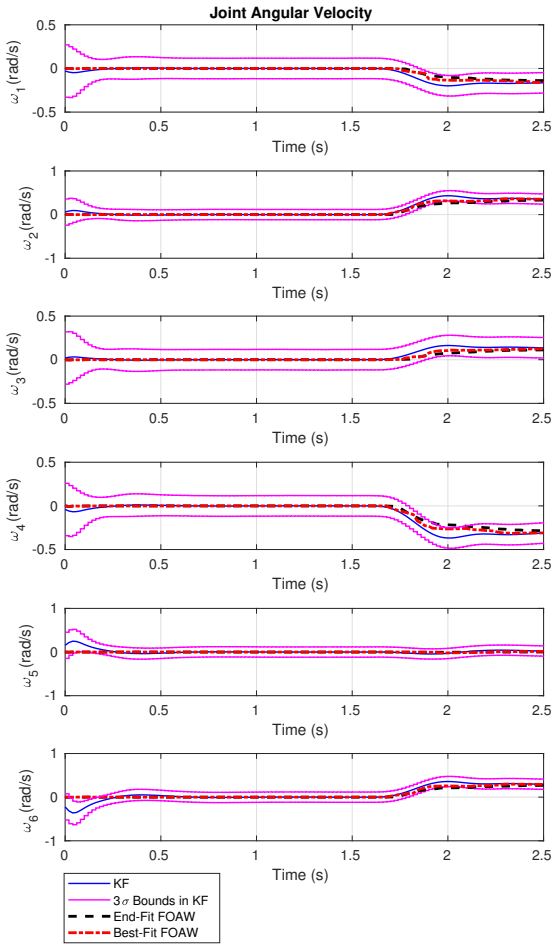

Fig. 6. Plots of the estimated angular velocity using Kalman filtering, end-fit adaptive windowing, and best-fit adaptive windowing during the first 2.5 seconds of the scenario. The 3σ bounds for the KF estimates are also provided.

On the contrary to the filtering approach, the adaptive windowing methods do not have any tuning parameters, as well as the covariance matrices of the errors. The only parameter affecting the performance of the estimator is the threshold of the error on which we decide whether the line is passing through the intermediate points. Here the threshold is set as

the maximum encoder error, as suggested by [2].

B. Performance Comparison

The three methods are compared from three main aspects: precision, time delay, implementation and practicality. Figures 5, 6 and 7 shows the plots of the first 2.5 seconds of the pick and place scenario.

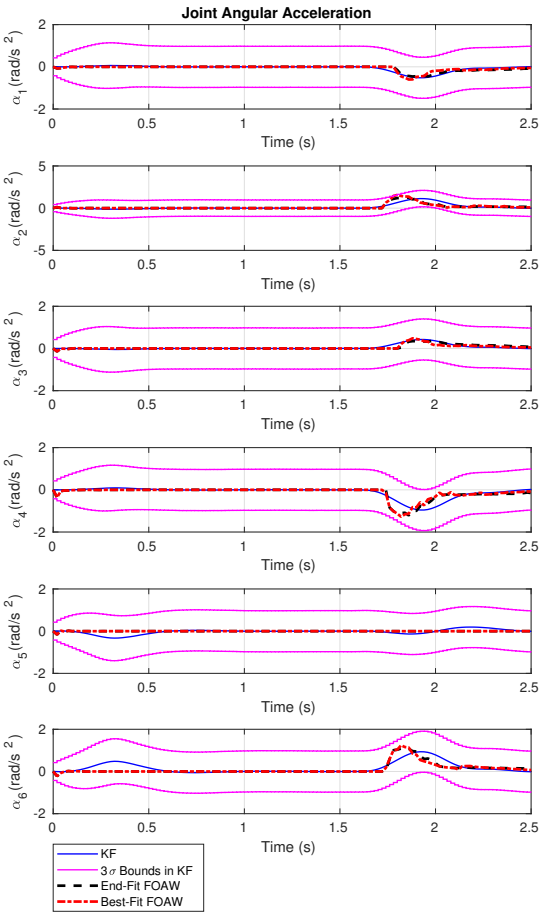

Fig. 7. Plots of the estimated angular acceleration using Kalman filtering, end-fit adaptive windowing, and best-fit adaptive windowing during the first 2.5 seconds of the scenario. The 3σ bounds for the KF estimates are also provided.

1) Precision: As the ground truth for the angular velocity and acceleration are not available, the accuracy of the estimates needs to be checked on a toy-problem consisting of a noise corrupted periodic function. The results obtained indicated that best-fit FOAW method outperforms end-fit FOAW and Kalman filter. Note that with $q = 0.01$, the precision of the Kalman filter is increased and it is as close as possible to the estimates of best-fit FOAW.

2) Time Delay: In terms of time delay, the Kalman filtering approach is more flexible compared to the two other methods. Depending on the application, one may choose larger values for q in order to decrease the time delay and increase the overshoot. However, as aforementioned, this is not plausible with the adaptive filtering method.

3) Implementation and Practicality: Unlike adaptive windowing, which is based on finite difference methods, Kalman filters provide the covariance of the estimates which may come in handy in many applications. Another advantage of using a Kalman filter over first-order adaptive windowing, is that it estimates position, velocity and acceleration, all in the form of a series of matrix multiplication. Thus, it can be used at high sampling rates. On the other hand, first-order adaptive windowing is a more computationally expensive method, as for each estimate it iterates over an inner loop to find out the best window size. Therefore, it will not be applicable for real time applications with high sampling rates.

VII. Conclusion

In this research, the two main approaches-filtering and adaptive windowing-were implemented on a ROS-running framework in order to estimate the velocity and acceleration in each joint. It was found that the adaptive windowing approach can provide a balance between precision and reliability. However, it is not able to provide any covariance for the estimates. Additionally it is not applicable to high sampling rates, due to its computationally expensiveness. On the other hand, filtering approach not only provides the certainty of the estimates, but also with an adjustment parameter, the performance of the system can be tweaked to account for time delay or overshoot. To conclude, the Kalman filtering approach proved to be superior to the adaptive filtering in estimation of the joint velocity and acceleration on a robotic manipulator.

References

[1] K. Robotics, "Jaco," *URL: http://kinovarobotics. com/products/jaco-robotics/(visited on 03/27/2015)*, 2013.

[2] F. Janabi-Sharifi, V. Hayward, and C.-S. Chen, "Discrete-time adaptive windowing for velocity estimation," *IEEE Transactions on control systems technology*, vol. 8, no. 6, pp. 1003–1009, 2000.

[3] A. Harrison and C. McMahon, "Estimation of acceleration from data with quantization errors using central finite-difference methods," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 207, no. 2, pp. 77–86, 1993.

[4] P. R. Belanger, "Estimation of angular velocity and acceleration from shaft encoder measurements," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. 585–592.

[5] P. R. Belanger, P. Dobrovolny, A. Helmy, and X. Zhang, "Estimation of angular velocity and acceleration from shaft-encoder measurements," *The International Journal of Robotics Research*, vol. 17, no. 11, pp. 1225–1233, 1998.

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[7] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

[8] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

Sahand Rezaei-Shoshtari Sahand Rezaei-Shoshtari was born in Tehran, Iran, in 1994. He received his B.E. degree in mechanical engineering form the University of Tehran in 2016. He is currently studying at McGill University, Montreal, Canada as a M.E. student. His current research focuses on the applications of machine learning and data-driven modeling in robotics and controls.