FPGA-Based Velocity Estimation for Control of Robots with Low-Resolution Encoders

Jie Ying Wu, Zihan Chen, Anton Deguet and Peter Kazanzides

***Abstract*— Robot control algorithms often rely on measurements of robot joint velocities, which can be estimated by measuring the time between encoder edges. When encoder edges occur infrequently, such as at low velocities and/or with low resolution encoders, this measurement delay may affect the stability of closed-loop control. This is evident in both the joint position control and Cartesian impedance control of the da Vinci Research Kit (dVRK), which contains several low-resolution encoders. We present a hardware-based method that gives more frequent velocity updates and is not affected by common encoder imperfections such as non-uniform duty cycles and quadrature phase error. The proposed method measures the time between consecutive edges of the same type but, unlike prior methods, is implemented for the rising and falling edges of both channels. Additionally, it estimates acceleration to enable software compensation of the measurement delay. The method is shown to improve Cartesian impedance control of the dVRK.**

I. Introduction

This paper presents a hardware-based velocity estimation method that enables improved control performance, especially for robots with low-resolution encoders and low gear reductions. The motivation for this work was provided by the da Vinci Research Kit (dVRK) [1], Fig. 1, which combines open source electronics and software with mechanical components of first-generation da Vinci surgical robots (Intuitive Surgical, Sunnyvale, CA). The da Vinci consists of a master console with two 7 degree-of-freedom Master Tool Manipulators (MTMs) and a patient side cart with several Patient Side Manipulators and an Endoscopic Camera Manipulator. This work focuses on the MTMs, which have wrist actuators with encoders with as few as 16 lines per revolution and gear ratios as low as 16.58.

The dVRK electronics relies on field-programmable gate arrays (FPGAs) to process the robot feedback, including quadrature decoding of the encoder signals, which are transfered to a control PC via IEEE-1394a (FireWire). The PC performs joint and Cartesian-level control at loop rates in excess of 1 kHz. The FPGA firmware (Verilog) and PC software (primarily C++) are open source. During system development, it was discovered that the standard proportional-integral-derivative (PID) joint controller had stability issues for the MTM wrist actuators. Specifically, the actuators had a tendency to shake (i.e., exhibit limit cycles). The dVRK actually uses PD control (no integrator) so the instability was traced to poor velocity estimation, which affected the derivative term. The problem was addressed by incorporating a heuristic nonlinear gain. Furthermore, the Cartesian impedance controller exhibited stability problems for the wrist actuators, especially the final roll axis, so impedance control was disabled for this actuator.

Dept. of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA {jieying, zihan.chen, anton.deguet, pkaz}@jhu.edu

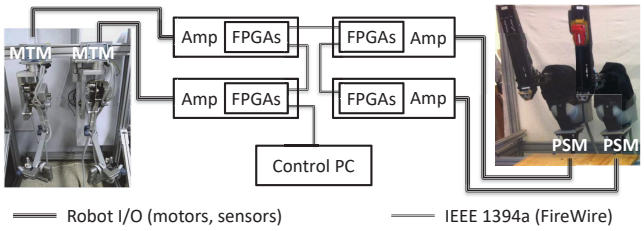


Fig. 1. da Vinci Research Kit: open source FPGA-based controllers connected to control PC via IEEE 1394a (FireWire), with direct access to motors and sensors in Master Tool Manipulators (MTMs) and Patient Side Manipulators (PSMs).

Section II-A provides an overview of methods for estimating velocity from quadrature incremental encoders. Fundamentally, these methods all rely on measuring the time between encoder position changes. Thus, measurements become more delayed as the joint velocity decreases. Robots with low-resolution encoders and low gear ratios experience larger delays because there are fewer counts per joint revolution. As discussed in Section II-A, this leads to a tradeoff between the responsiveness of the velocity estimation and its robustness to imperfections that introduce noise. Unfortunately, noisy or delayed measurements negatively affect control performance, including for the Cartesian impedance controller presented in Section II-B.

In Section III, we propose a novel hardware-based method that improves the responsiveness of the velocity estimation by using all encoder edges and by also estimating the acceleration. This enables the PC software to compensate for the measurement delay, leading to a solution that provides timely and robust velocity estimates. Section IV first evaluates the method on a test platform with two mechanically-coupled encoders (one high resolution and one low resolution) and then with the Cartesian impedance controller on the dVRK. The significance of this work is that it improves the control performance of the dVRK, a common research platform currently installed at 30 institutions worldwide.

II. Background and Related Work

A. Velocity Estimation with Quadrature Encoders

Many systems (including the da Vinci) estimate velocity from quadrature incremental encoders, which contain two channels (A and B) that produce square waves that are 90 degrees out of phase (Fig. 2). We consider each of the

following as separate events: 1) rising edge of the A channel, $A_{\uparrow}$, 2) falling edge of the A channel, $A_{\downarrow}$, 3) rising edge of the B channel, $B_{\uparrow}$, and 4) falling edge of the B channel, $B_{\downarrow}$. The joint position is obtained by counting each of these edges, with the direction determined by identifying which channel leads the other.

Velocity is estimated by calculating dx/dt, where dx is the encoder position difference and dt is the sampling interval (i.e., time between the two encoder position measurements). Typically, dx or dt is fixed to obtain either a fixed-time or a fixed-position algorithm, though some variations, such as the constant elapsed time (CET) method [2], measure time over multiple counts to achieve a minimum elapsed time.

In cases where the velocity estimation module does not have direct access to the encoder edges, it is only feasible to use a fixed-time approach, where the time between the two position samples is based on the CPU clock. This reduces the accuracy of the time measurement because it is not synchronized with respect to the position updates. If, however, the module has access to the encoder edges, either method may be used. For the fixed-position methods, measuring the time between two edges of the same type (full-cycle measurement, as shown in Fig. 2) increases the measurement delay but is robust to imperfections in the encoder phase (i.e., channels not exactly 90 degrees apart) and the duty cycle (i.e., channel high times not exactly equal to low times at constant velocity). In contrast, measuring the time between the two most recent edges (a quarter-cycle measurement) leads to noisy velocity measurements, where much of the noise is due to these encoder imperfections. At high speeds, the fixed-position methods suffer from time quantization errors because fewer clock ticks occur between the encoder edges (small dt), whereas at low speeds the fixed-time methods are subject to position quantization errors (small dx) and are also more affected by encoder imperfections.

Brown *et al.* compare the fixed-position and fixed-time methods for different velocity profiles as well as add higher-order terms on each scheme [3]. Most fixed-position algorithms tested were sensitive to encoder imperfections, while fixed-time algorithms were sensitive to quantization errors at low speeds. Adding higher order terms through Taylor series expansion and backward-difference expansion exaggerated these errors. While least-square-fit smooths over imperfections and quantization errors, it has bad transient response because it acts as a filter.

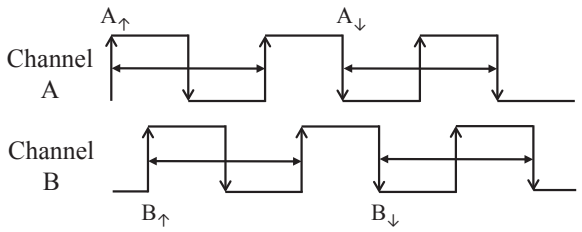


Fig. 2. Quadrature incremental encoder feedback, showing four events ($A_{\uparrow}$, $A_{\downarrow}$, $B_{\uparrow}$, and $B_{\downarrow}$) and time between consecutive occurrences of the same event.

Low-resolution encoders and low gear ratios introduce challenges to the above methods, especially at low velocities due to the longer delays between encoder edges. We focus on fixed-position algorithms as they perform better when there is long delay between signals [4] and, in contrast to the results reported by Brown *et al.* [3], they can be implemented to be insensitive to common encoder imperfections. In practice, the definition of "low" velocity depends on the gear ratio that relates motor/encoder revolutions to robot joint revolutions. Sakata and Fujimoto use a plant model of the motor dynamics to overcome the inherent delay where the velocity measurement is always half a cycle behind due to the nature of averaging over a period [5]. To overcome delays in measurement without knowledge of motor specifications, Nandayapa *et al.* propose to add fractional steps to the position measurements in between encoder events and show that using this position for velocity estimation gives more stable results [6].

Similarly, model and non-model based methods have been examined to provide robustness against noise. Model-based algorithms can provide more accurate estimates, but require a dynamic model and a noise model, which are often inaccurate or unavailable. Zhu and Sugie [7] show that by using polynomial fitting on position data with knowledge of motor dynamics, they can accurately track the velocity for encoders with as low as 8 pulses per revolution. Kim and Kim [8] reduce this computational load by fitting on the sparser transition edges of the encoder. Non-model based algorithms are often filter-based and introduce delays. Merry *et al.* [9] examine some trade-offs in filter length and propose additional parameters such as skipping edges so filters can examine a sufficient length of data without being too memory-intensive.

The proposed method differs from the above methods first by using the rising and falling edges of both encoder channels for velocity estimation. This improves the responsiveness of the velocity estimation by a factor of four over the conventional method of measuring time between one type of edge on one encoder channel, while preserving that method's robustness to encoder imperfections. We consider only the fixed-position method because for the dVRK, the encoder resolutions and the maximum speeds with a human operator are low enough that time quantization error is not a significant issue. Because this velocity estimation still has larger delays than the quarter-cycle difference, the method additionally estimates acceleration. This enables the PC software to compensate for the measurement delay, leading to a solution that provides both the responsiveness of the quarter-cycle measurement and the robustness of the full-cycle measurement. Note that since the goal of the proposed method is to provide the best information possible from hardware, any of the above software algorithms could be adopted to further improve results.

B. Cartesian Impedance Control

Figure 3 shows the implementation of the dVRK Cartesian impedance controller. The torque applied to the robot is calculated similarly to the method described in [10], [11].

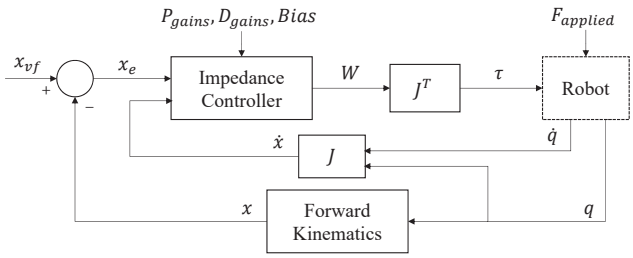


Fig. 3. Block diagram of dVRK Cartesian impedance controller

This avoids calculating the inverse Jacobian by using the Jacobian transpose to calculate the torque applied to each joint.

In the figure, $(q, \dot{q})$ are the robot state (joint position and velocity) as measured by the encoders. This is converted into Cartesian position x and velocity $\dot{x}$ by forward kinematics. A force is applied to the robot based on the error in position, where the tool tip is compared to the virtual fixture position, x_{vf}, and the velocity feedback provides damping. A constant bias force is also applied. The impedance controller then calculates the Cartesian wrench, *W*, for translation along, and rotation around, each of the three axes as follows:

$$W = P\left(x - x_{vf}\right) - D\dot{x} + Bias \tag{1}$$

P and D are the proportional and derivative gains and $Bias$ accounts for constant offsets such as gravity compensation.

III. Method

A. Full-cycle Velocity Estimation

We observe that the velocity calculated over a quarter cycle is noisy, in large part due to the effect of encoder imperfections, and therefore calculate it over a full encoder cycle. We measure the time from one instance of an event to the next instance of the same event, as illustrated in Figure 2. Setting the cycle time as S and using i to indicate the event, the velocity over a full cycle (4 encoder events) can be calculated by:

$$v_i = \frac{4}{S_i} \tag{2}$$

B. Acceleration Estimation

While Equation (2) gives a smooth velocity estimation, the measurements are delayed because the data is estimated over a full quadrature cycle. Thus, a common approach is to assume that the currently measured velocity corresponds to the velocity in the middle of the cycle; i.e., it has a delay of a half cycle ($S_i/2$). We encountered significant delays at low velocities when calculating velocity over a full quadrature cycle compared to that of the quarter-cycle, which led to unstable controls. To estimate the velocity change over the last half cycle, we add an acceleration term, a:

$$v_i = \frac{4}{S_i} + a\frac{S_i}{2} \tag{3}$$

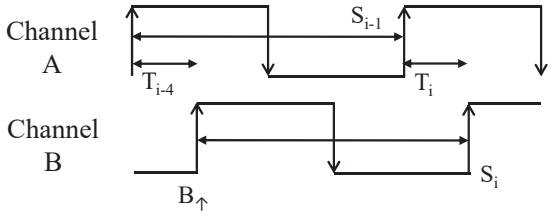


Fig. 4. Illustration of acceleration estimation. S indicates full cycles, which are robust to encoder imperfections, while T indicates quarter cycles, which are not. Acceleration can be estimated by the change between quarters of the same type, T_{i-4} and T_i, to be robust to encoder imperfections. In this example, both quarters are measured from $\mathrm{A}_\uparrow$ to $\mathrm{B}_\uparrow$.

The predicted velocity at i is the velocity over S_i plus acceleration over $S_i/2$, assuming constant acceleration. We estimate acceleration as a backward difference between the last two full-cycle velocity measurements. Effectively, this leads to subtraction of two quarter-cycle events that are separated by a full cycle. For example, if the last two events were $\mathrm{A}_\uparrow$ followed by $\mathrm{B}_\uparrow$ (as shown in Figure 4), the acceleration would be calculated from the difference between the last quarter and quarter that happened 4 events ago, both of which would be between $\mathrm{A}_\uparrow$ and $\mathrm{B}_\uparrow$, assuming no direction change. As with the velocity estimation, comparing the time between two events of the same type avoids the effects of encoder imperfections such as uneven duty cycles and phase shifts, which are a primary cause of measurement noise.

Mathematically, the change in velocity over the last two cycles, S_i and S_{i-1}, is given by:

$$\begin{aligned} \Delta v =& \frac{4}{S_i} - \frac{4}{S_{i-1}} \\ \Delta v =& \frac{4(T_{i-4} - T_i)}{S_i S_{i-1}} \end{aligned} \tag{4}$$

The time difference between these two velocities (assumed to correspond to the midpoints of the cycles) is given by the following, where t_i indicates the time at event i:

$$\begin{aligned} \Delta t =& \left(t_i - \frac{S_i}{2}\right) - \left(t_i - T_i - \frac{S_{i-1}}{2}\right) \\ \Delta t =& \frac{T_{i-4} + T_i}{2} \end{aligned} \tag{5}$$

Finally, the instantaneous acceleration is simply the quotient between the change in velocity, Δv, and time, Δt:

$$a = \frac{\Delta v}{\Delta t} = \frac{8(T_{i-4} - T_i)}{S_i S_{i-1}(T_{i-4} + T_i)} \tag{6}$$

Substituting Equation (6) into Equation (3) yields:

$$\begin{aligned} v_i =& \frac{4}{S_i} + \frac{8(T_{i-4} - T_i)}{S_i S_{i-1}(T_{i-4} + T_i)} \frac{S_i}{2} \\ v_i =& \frac{4}{S_i} + \frac{4(T_{i-4} - T_i)}{S_{i-1}(T_{i-4} + T_i)} \end{aligned} \tag{7}$$

Lastly, for the most up-to-date estimate, we can add a running counter, T_r, that measures the time since the last event:

$$v_i = \frac{4}{S_i} + \frac{8(T_{i-4} - T_i)}{S_i S_{i-1}(T_{i-4} + T_i)} \left(\frac{S_i}{2} + T_r \right) \quad (8)$$

Although this method avoids the noise due to encoder imperfections of duty cycle and phase, it is sensitive to quantization error because the difference between quarters is often only a few counts. This can be avoided by setting a threshold that prevents acceleration from being used at high velocities. Quantization error increases linearly with velocity as the number of counts decreases. But, delays in velocity are more significant at slow speeds, where large quantization is not a significant concern.

C. Implementation

To keep track of the quarter measurements, we use a queue of six 26-bit registers (Figure 5). The first register is a running counter that is incremented at every clock edge. The clock speed is 49.152 MHz so the counter overflows in 1.37 s. Each edge of any type adds an element to the queue, pops off the last element, and clears the first counter. Thus, the first, second, and sixth register values are used to calculate the acceleration as they represent T_r, T_i, and T_{i-4}, respectively. S_i can be calculated from summing the quarter counters T_i to T_{i-3}.

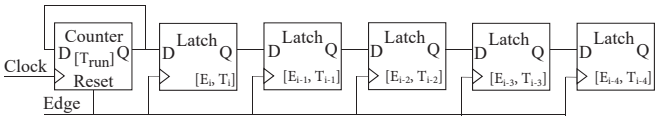


Fig. 5. Queue of running counter and quarter-cycle time measurements, $T_i \ldots T_{i-4}$, for edges $E_i \ldots E_{i-4}$.

The host computer issues asynchronous read requests on the FireWire bus to obtain feedback data from the FPGAs at a specified rate, generally 2-3 kHz. In the current implementation, S_i is sent as a 22-bit value, where the last 4 bits are truncated to give an effective count rate of 3.072 MHz (49.152/16). T_{i-4} and T_i are passed back as 20-bit values. Using 26-bit registers on the FPGA provides robustness against encoder imperfections because an uneven duty cycle can cause one quarter cycle to be more than 20-bits long, but the sum of four quarter cycles should still fit within 26-bits. On the PC, S_{i-1} is calculated from $S_{i-1} = S_i + (T_{i-4} - T_i)$.

We set a threshold to stop using acceleration if T_i is smaller than 2000 clock counts as it oscillates too much beyond that point. The exact threshold in velocity varies due to encoder imperfections and uneven rotation speeds.

Currently, the running counter, T_r, is not separately provided to the PC due to implementation limitations and thus Equation (7) is used. As a consolation, if T_r is greater than T_{i-4} the FPGA uses T_r instead of T_{i-4} when calculating S_i. This provides smoother decays in cases where the motor is decelerating, as would be obtained by using T_r in Equation (8).

IV. Experimental Setup

We show the proposed method implemented on two hardware setups. In the first, two encoders, one high-resolution and one low-resolution, are coupled and driven by a DC motor. The gearing of the motor was removed so that the two encoders rotate at the same rate. In the second, we show the proposed method implemented on the dVRK and test its Cartesian impedance control with different velocity estimation methods.

A. Two Encoder Setup

Two encoders are connected as shown in Figure 6. The left encoder has 600 lines per revolution and we use this as the ground truth position and velocity data. The right one has 16 lines per revolution and is attached to a motor.

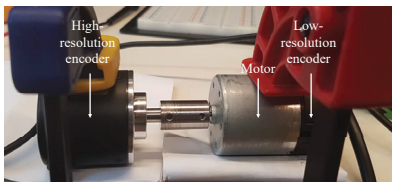


Fig. 6. Coupled encoders

To test the proposed method, we drive the motor and estimate the velocity by the proposed method, with and without acceleration, and compare it to velocity estimated by the quarter-cycle method (time between last two position changes). In our experiments, we never exceed one count difference per sampling time on the low-resolution encoder, so fixed-time algorithms are not considered.

B. Cartesian Impedance Control

The effect of the proposed velocity estimation on control performance was tested on the left MTM of the dVRK (Figure 7), where the resolution of encoders ranges from 1000 lines (4000 counts) per revolution in the first four joints to 16 lines (64 counts) per revolution in the last three (wrist)

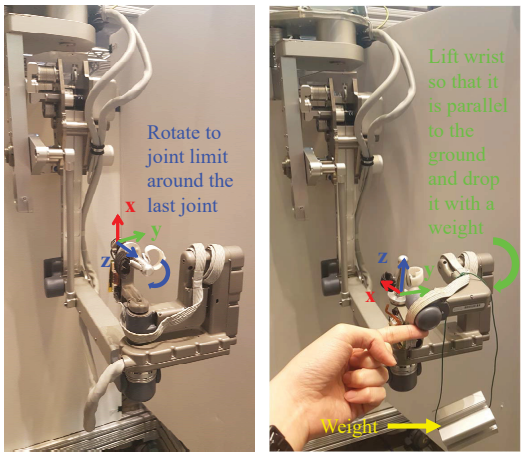
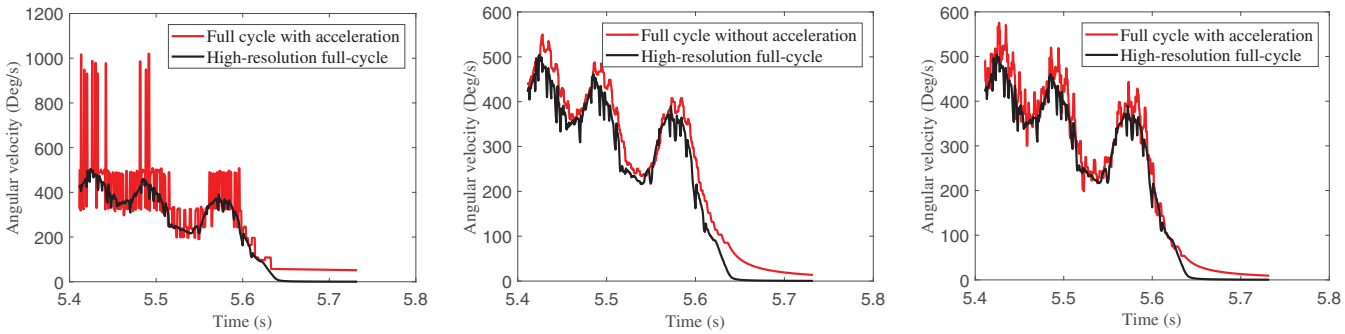


Fig. 7. Master controller of the dVRK robot. The left image shows the joint that we rotate around the z-axis, while the right image shows the wrist being lifted around the y-axis, with a weight attached.



(a) Quarter-cycle estimation (b) Full-cycle estimation without acceleration (c) Full-cycle estimation with acceleration

Fig. 8. Comparing velocity estimation for 16 lines low-resolution encoder (red) with 600 lines high-resolution encoder (black) for different velocity estimation methods: (a) quarter cycle oscillates at high velocities, (b) full cycle is noticeably delayed even at high velocities, and (c) full cycle with acceleration (proposed method) reduces the delay but has some oscillations.

joints. Furthermore, the gear ratios of the last three joints are 33.16, 33.16, and 16.58, which lead to resolutions of 0.17 deg/count, 0.17 deg/count and 0.34 deg/count, respectively.

We perform two experiments, where the first primarily exercises the last wrist joint and the second excites the larger joints and the first wrist joint. In each experiment, the robot is moved to a consistent home position and then a horizontal plane virtual fixture is created to prevent the arm from dropping due to gravity. For the first experiment, the desired (virtual fixture) orientation is set to the home orientation. The robot is set to Cartesian impedance mode and we manually rotate the last joint clockwise around the z-axis until it reaches its limit, as shown in Figure 7-left. We then release it and measure the step response as the robot recovers the desired orientation. Cartesian impedance gains were set at 200 N/m and 5 for linear stiffness and damping, and 0.15 N m/rad and 0.03 for torsional stiffness and damping.

In the second experiment, we set the desired orientation so that the first wrist link (closest to the robot base) is rotated by 90° around the y-axis, as shown in Figure 7-right. A weight of 121.5 g is attached to the wrist link while holding it in the desired orientation. We then release the link and measure the robot's position and orientation as the Cartesian impedance controller finds a new equilibrium. The two most distal wrist joints are not affected by the applied weight, and because they are orthogonal to the one supporting the weight, their possible motion has no effect on the results. We use the same impedance gains as before, except that the torsional damping is set to 0.04 to prevent the weight from hitting another link.

V. Results

A. High and Low Resolution Encoder Comparison

Figures 8a to 8c show velocity estimated by different methods on the low-resolution encoder compared to a high-resolution ground truth. The high-resolution velocity is calculated from the full cycle, without acceleration, since the measurement delay is small and thus does not require compensation based on acceleration. The RMS error between the velocity estimated by the low-resolution encoder and the ground-truth is 44.12 deg/s for the quarter-cycle method, 23.49 deg/s for the full-cycle method without acceleration, and 19.48 deg/s for the full-cycle method with acceleration. Without acceleration, we observe from Figure 8b that measurements, even at fairly high velocities, are noticeably delayed, and that the delay gets worse closer to 0. While the quarter-cycle velocity estimation in Figure 8a is the least delayed, it has large oscillations (noise) at high velocities from one-pulse differences or pulse-alterations as also observed in [12]. The proposed method in Figure 8c obtains a reasonable compromise between delay and noise and has the lowest error with respect to the ground truth signal.

B. Cartesian Impedance Control

Figure 9 shows the step responses when the last joint is released after having been rotated to its limit. The quarter-cycle velocity estimation never settles, while the full-cycle method has a delayed response. The proposed method (full-cycle with acceleration) is initially noisy, but settles the fastest. While the proposed method has more oscillations, in the dVRK, its shorter settling time resulted in better control performance.

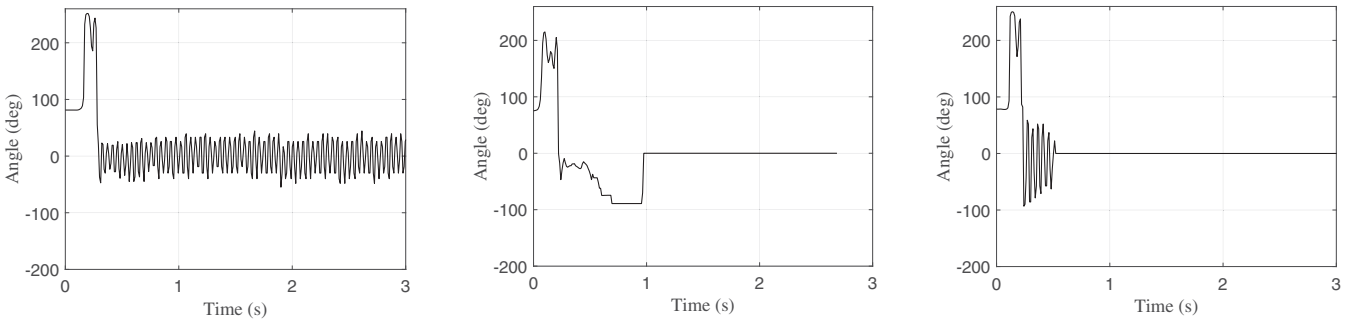
Figure 10 shows the step response when a weight is dropped on the third-to-last wrist joint. All methods provide stable performance because this experiment does not involve the lowest resolution joint, but the full-cycle estimation without acceleration is the slowest to respond, as expected.

Table I summarizes the means and standard deviations of the settling times over 10 trials for displacing the wrist around the z and y axes, respectively. While not demonstrated in these experiments, the proposed velocity estimation method also improves joint control performance with a conventional PD controller (i.e., without the heuristic nonlinear gain).

TABLE I

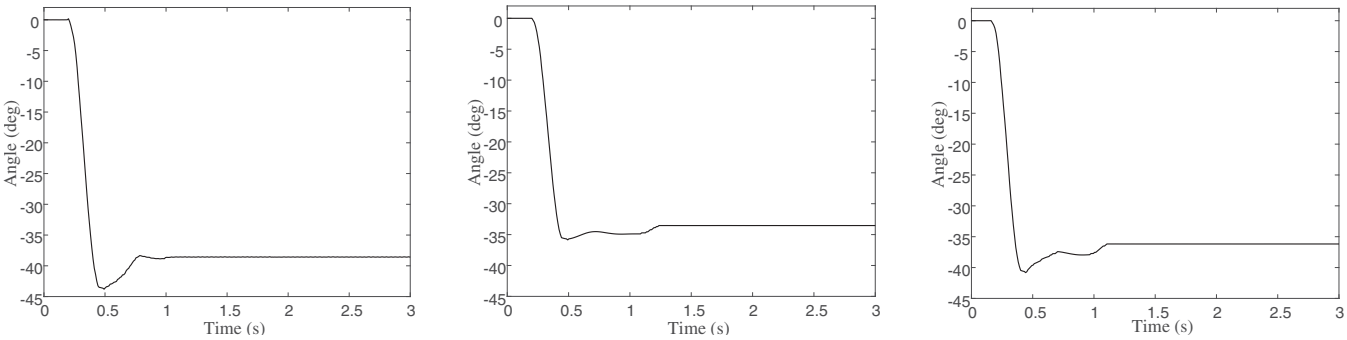
Mean and standard deviation of settling times of rotation around z and y axes over 10 trials.

Settling time (s)	Quarter cycle	Full cycle	
		Without acc	With acc
Rotation in z	-	1.05 ± 0.62	**0.91 ± 0.31**
Rotation in y	1.76 ± 0.52	2.05 ± 0.49	**1.65 ± 0.41**



(a) Quarter-cycle estimation (b) Full-cycle estimation without acceleration (c) Full-cycle estimation with acceleration

Fig. 9. Response of Cartesian impedance controller when torque applied and released about z-axis, which primarily displaces a low-resolution, low-gear ratio joint.



(a) Quarter-cycle estimation (b) Full-cycle estimation without acceleration (c) Full-cycle estimation with acceleration

Fig. 10. Step response of Cartesian impedance controller when adding a weight (moment) around y-axis.

VI. Conclusions

This work proposes a novel method to estimate velocity using all edges of a quadrature incremental encoder and using acceleration to overcome delays in measurement to provide better feedback for closed-loop control. The results show that this method produces a smoother, more accurate and timely velocity estimate on a low-resolution encoder. Furthermore, we showed that the improved velocity estimate leads to more stable control of a robot that has low-resolution encoders, with as few as 16 lines per revolution. The implementation (FPGA firmware and C++ software) is available open source and improves the performance of the dVRK, currently installed at 30 institutions worldwide. In addition, the proposed technique could be combined with model-based velocity estimation to further improve its accuracy, especially when a motor begins moving and there is insufficient data to calculate acceleration.

VII. Acknowledgement

This work is supported by NSF NRI 1637789.

References

[1] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da Vinci® Surgical System," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 6434–6439.

[2] R. Bonert, "Digital Tachometer with Fast Dynamic Response Implemented by a Microprocessor," *IEEE Transactions on Industry Applications*, vol. IA-19, no. 6, pp. 1052–1056, 1983.

[3] R. H. Brown, S. C. Schneider, and M. G. Mulligan, "Analysis of algorithms for velocity estimation from discrete position versus time data," *IEEE Trans. on Industrial Elect.*, vol. 39, no. 1, pp. 11–19, 1992.

[4] N. Ekekwe, R. Etienne-Cummings, and P. Kazanzides, "Incremental encoder based position and velocity measurements VLSI chip with serial peripheral interface," in *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, 2007, pp. 3558–3561.

[5] K. Sakata and H. Fujimoto, "Proposal of long sampling short cycle observer for quantization error reduction," in *IEEE Intl. Symp. on Industrial Electronics (ISIE)*, 2010, pp. 1919–1924.

[6] M. Nandayapa, C. Mitsantisuk, and K. Ohishi, "High resolution position estimation for advanced motion control based on FPGA," in *38th Annual Conf. of IEEE Industrial Elect. Society (IECON)*, 2012, pp. 3808–3813.

[7] H. Zhu and T. Sugie, "Velocity estimation of motion systems based on low-resolution encoders," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, no. 1, pp. 011 006:1–8, 2013.

[8] J. Kim and B. K. Kim, "Development of precise encoder edge-based state estimation for motors," *IEEE Trans. on Industrial Electronics*, vol. 63, no. 6, pp. 3648–3655, 2016.

[9] R. J. E. Merry, M. J. G. van de Molengraft, and M. Steinbuch, "Optimal higher-order encoder time-stamping," *Mechatronics*, vol. 23, no. 5, pp. 481–490, 2013.

[10] N. Hogan, "Impedance Control: An Approach to Manipulation: Part II - Implementation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, p. 8, 1985.

[11] A. Albu-Schäffer and G. Hirzinger, "Cartesian impedance control techniques for torque controlled light-weight robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2002, pp. 657–663.

[12] T. Tsuji, T. Hashimoto, H. Kobayashi, M. Mizuochi, and K. Ohnishi, "A wide-range velocity measurement method for motion control," *IEEE Trans. on Industrial Electronics*, vol. 56, no. 2, pp. 510–519, 2009.