Comparison of LTI and Event-Based Control for a Moving Cart with Quantized Position Measurements

Henningsson, Toivo; Cervin, Anton

2009

Comparison of LTI and Event-Based Control for a Moving Cart with Quantized Position Measurements

Toivo Henningsson and Anton Cervin

***Abstract*— Traditional linear time-invariant (LTI) control design assumes that measurements are taken at regular time intervals and have independent additive noise. A common practical case that violates this assumption is the use of encoders that give quantized position measurements; when the quantization is appreciable the measurement noise is far from LTI. This paper develops a simple event-based controller based on simplifying a joint maximum a posteriori estimator, which is applied to a moving cart with quantized position measurements. The payoff for implementing the somewhat more complex event-based controller is to drastically reduce the effect of quantization noise in the experiments. A sequence of simpler to better adapted controllers are described and compared according to experimental performance and implementation complexity.**

I. Introduction

The majority of all feedback controllers today are implemented using computers, relying on periodic sampling, computation, and actuation. For linear time-invariant (LTI) systems, sampled-data control theory [3] provides powerful tools for direct digital design, while implementations of nonlinear control designs tend to rely on discretization combined with fast periodic sampling.

There are however situations where it could be advantageous to use other activation schemes. For first-order linear stochastic systems, it has been shown that event-triggered sampling can provide better regulation performance and/or lower average activation rates than time-triggered sampling [2], [7]. This can be useful for networked embedded control systems with constrained communication, computation, or energy resources. With similar arguments, heuristic event-based PID controllers have been proposed in [1], [10].

Another motivation for event-based control are systems where the events are inherent in the physics. Examples include wheel encoders and accelerometers that deliver pulse trains rather than continuous measurement signals. Previous case studies have shown that accurate control can be accomplished even with very low-resolution encoders if the controller is activated at measurement events rather than at regular time intervals [9].

In this paper, we study the practical problem of implementing a velocity control system for a moving cart using a low-resolution position encoder and a low-end 8-bit microcontroller. Each time the quantized position measurement changes value is considered an *event*, to be given special consideration by the controller. Since the friction is appreciable and varying, we want to utilize also the information contained in *the absence of events*.

It is well known that the problem of optimal estimation and control with quantized measurements is extremely difficult [5]. For instance, there is no separation theorem in the general case. Even the pure estimation problem is computationally intractable and essentially requires the online solution of a partial differential equation.

Seeking simpler, sub-optimal solutions, we start from a joint maximum a posteriori (JMAP) estimator [4]. This formulation is powerful enough to model quantized measurements, yet yields a tractable problem. The estimator is greatly simplified and adapted to the control problem at hand in order to be implementable on the small microcontroller. The final controller is based on periodic state feedback from an event-based observer implemented using fast sampling. The state feedback design can be reused from the LTI controllers designed for comparison, since the practical challenge lies in state estimation.

The rest of the paper is laid out as follows. The setup is explained in Section II. Section III attempts a first control design using LTI methods, giving important insight into the tradeoffs involved. The JMAP estimator is introduced in Section IV as a systematic means for state estimation with quantized measurements and is simplified in Section V into something that can run online. Microcontroller implementation issues are described in Section VI. Section VII compares different LTI and event-based controllers experimentally. The conclusions are given in Section VIII.

II. Setup

A. The Moving Cart

The process is a moving cart driven by a DC motor. The control signal is the motor voltage, governed by a direction bit and a 29 kHz PWM signal. A rotary encoder on the motor axis is used for position sensing. The encoder output is two square waves as a function of the position, 90° out of phase. The measurements can be modeled as

$$y = \Delta p_{\text{quant.}} \cdot \text{round}\left(\frac{p}{\Delta p_{\text{quant.}}}\right), \; \Delta p_{\text{quant.}} = 5 \cdot 10^{-5}\,\text{m}, \quad (1)$$

where p is the position of the cart. Lower encoder resolutions can be emulated in software; most tests are run with $\Delta p_{\text{quant.}}$ emulated to 3.2 mm.

The cart is equipped with an ATmega16 8-bit AVR microcontroller clocked at 14.7 MHz, which handles encoder sampling, motor drive, filtering, control, and communication with a PC over a serial link.

T. Henningsson and A. Cervin are with the Department of Automatic Control LTH, Lund University, Box 118, SE-221 00 Lund, Sweden. Email: `toivo.henningsson@control.lth.se`

B. Process Model

A simple dynamical model for the cart was postulated as

$$\begin{pmatrix}\dot p \\ \dot v\end{pmatrix} = \begin{pmatrix}0 & 1 \\ 0 & -T_d^{-1}\end{pmatrix}\begin{pmatrix}p \\ v\end{pmatrix} + \begin{pmatrix}0 \\ 1\end{pmatrix}(u + u_{\text{bias}}),$$

where p is the position, v is the velocity, $u = K_u u_V$ is the control signal, u_V is the control signal in units of full motor voltage, u_{bias} is the disturbance from friction and other sources, and K_u and T_d are process parameters. A sequence of step response experiments were made and the parameters estimated by linear regression to $T_d = 0.23\,\text{s}$, $K_u = 25.5\,\text{m/s}^2$. The disturbance u_{bias} was assumed constant within each step response.

As the controllers to be designed will have cross-over frequency $\omega_c \geq 5$ rad/s, T_d^{-1} is approximated to 0 for simplicity. With this approximation the process is a pure double integrator, $G_p(s) = \frac{1}{s^2}$, and the cross-over frequency of a control design can be adjusted by just changing the time scale. The time scale of the process is adjusted by additional gain in the controller.

C. Control Objectives

We will design velocity controllers for the moving cart. The objectives are:

- Fast reference tracking and attenuation of process disturbances (friction).
- Low noise in the control signal.
- Reasonable robustness.

Reasonable robustness is an absolute demand. Given this constraint, the controllers should try to optimize for the first two objectives, using the cross-over frequency ω_c to adjust the tradeoff.

D. Implementation Structure

A multi-rate structure is used, where fast sampling at 100 kHz is used to read the encoder, while the control output is generated at 1 kHz. For simplicity, the implementation is not event-triggered *per se*; rather, the estimator may use time stamps of the latest measurement events (i.e., changes in encoder value) when forming its estimate.

III. LTI Control Design

In this section, we design two PI controllers under simplifying assumptions. The design is carried out in continuous time, ignoring the fact that the measurement disturbance comes from quantization. Still, the design provides important insight into robustness issues.

A. Pure PI Controller

To do velocity control, we need to estimate the cart velocity. We use a simple first-order filter on the position output:

$$\hat V = \frac{sY}{sT_{\text{filter}} + 1}. \quad (2)$$

A PI controller

$$U = K(\beta V_{\text{ref}} - \hat V) - K\frac{\hat V - V_{\text{ref}}}{sT_i} \quad (3)$$

gives the following controller transfer function from $-y$ to u:

$$G_c(s) = \frac{K}{T_i}\frac{sT_i + 1}{sT_{\text{filter}} + 1}$$

This is in effect a lead filter that lifts the phase of the loop gain around the cross-over frequency ω_c above the constant phase $G_p(i\omega) = -180°$ of the process.

We want to place the zero and pole far apart to get a large phase margin, but on the other hand we want short T_i and long T_{filter} for good rejection of process disturbances and measurement noise. The best trade-off is achieved by placing the zero and pole on either side of the cross-over frequency ω_c at equal logarithmic distance,

$$T_{\text{filter}} = r^{-1}\omega_c^{-1}, \quad T_i = r\omega_c^{-1},$$

where $r > 1$ determines the phase margin ϕ_m. We take $\phi_m \approx 50° \implies r \approx 3$ as a reasonable compromise between robustness and disturbance rejection.

The cross-over frequency ω_c is left as a design parameter, to be varied in the experiments. The reference weighting parameter $\beta = 0.5$ is used to eliminate overshoot in the reference step response.

B. Observer-Based PI Controller

The standard PI controller makes no use of the process model. To exploit our process knowledge, we can instead use the control law

$$U = \underbrace{K(V_{\text{ref}} - \hat V)}_{U_P} - \underbrace{K\frac{sY - \hat V}{sT_i}}_{\hat U_{\text{bias}}}, \quad (4)$$

where the I-part now integrates only the difference between actual and estimated velocity, and the velocity estimate

$$\hat V = \frac{sY + T_{\text{filter}}U_P}{sT_{\text{filter}} + 1} \quad (5)$$

includes feedforward from the P-part. Since the I-part provides the bias estimate, it should not be fed forward into $\hat V$. Reacting only to differences between the model prediction and measurements, the I-part will no longer wind up during reference steps, which eliminates the need for the β tuning parameter. Using $U + \hat U_{\text{bias}}$ instead of U_P for feedforward in the velocity filter, this disturbance estimation scheme also provides anti-windup.

With the observer-based control law, the controller transfer function becomes

$$G_c(s) = \frac{K}{T_i}\frac{s(T_i + T_{\text{filter}}) + KT_{\text{filter}}}{sT_{\text{filter}} + 1 + KT_{\text{filter}}},$$

i.e., a lead filter with slower zero and faster pole than for the pure PI controller. The observer gives extra phase lead around the cross-over frequency, which can be exploited by more aggressive tuning. We take

- $T_i' = T_i/4$ for improved disturbance rejection.
- $T_i' = T_i/2, T_{\text{filter}}' \approx 2T_{\text{filter}}$ for improved measurement noise rejection. This is useful for the PI controller since it is bad at handling the measurement quantization. We

do not want to make T_{filter} slower than ω_c^{-1}, since this would impede process disturbance rejection.

Since the transfer functions $G_c(s)$ have the same form, the pure and observer-based PI controllers can be tuned to more or less the same behaviour in closed loop. The real gain of using an observer will be evident when we introduce the event-based controller, where the control runs in open loop as long as the measurements do not contradict the observer's predictions.

IV. The JMAP Estimator

A problem with the PI controller is that it does not exploit the fact that the main source of measurement noise comes from quantization. In this section, we explore how to model quantization in the state estimator, and the properties that follow. Insight into the behavior of the estimator will be used to simplify it in the next section.

A. Process Model

Consider a system in discrete time,

$$x(k+1) = Ax(k) + Bu(k) + w(k), \tag{6}$$

where x is the state, u is the control signal, and w is a zero-mean white Gaussian noise process with variance R. The available measurements specify an interval for the output at each sample:

$$y(k) - \Delta y \le Cx(k) \le y(k) + \Delta y. \tag{7}$$

The initial state may be fully known or Gaussian distributed.

B. The Estimation Problem

We consider the joint maximum a posteriori (JMAP) approach to state estimation: find the most probable trajectory of the state x conditioned on the measurements, and use the state at the current time as state estimate. With additive Gaussian measurement noise, this approach yields the Kalman filter (see [4]). With the quantized measurements (7), the solution is a bit more complex.

The log-likelihood $l(x)$ of a state trajectory $x(k), k_0 \le k \le k_1$, considering the dynamics (6) and initial distribution of the state is given by

$$l(x) = -\tfrac{1}{2}\left(||x(k_0) - x_0||^2_{R_0^{-1}} + \sum_{k=k_0}^{k_1-1} ||w(k)||^2_{R^{-1}} \right), \tag{8}$$
$$w(k) = x(k+1) - Ax(k) - Bu(k),$$

when $x(k_0)$ has a Gaussian distribution with mean x_0 and variance R_0, and where $||x||^2_R = x^T Rx$ and $w(k)$ has been solved for from (6). The most probable state trajectory x is found by maximizing $l(x)$ subject to the linear measurement constraints (7) and any known initial conditions. This is a quadratic program, which can be solved reasonably fast on a PC. Ideally, the history $k_0 \le k \le k_1$ should go as far back as possible to use all measurements in the estimation. An approach often used in practice is to fix $k_1 - k_0 = \Delta k$, resulting in *moving horizon estimation*, see [8].

At any sample k, the constraint (7) is considered *active* if the optimal trajectory would be different without it. If it is known which constraints are active, the optimal solution can be obtained by fixing the constrained variables at their constraints and optimizing freely over the rest. This is the same solution as we would get from a Kalman filter when the available measurements are perfect position measurements at the active constraints.

C. Time Update

When moving forward one sample to $k = k_1 + 1$ without adding new measurements, the estimate update is very simple. Suppose that there is a unique optimal state trajectory. The new cost term in (8) will add no cost iff $w(k_1) = 0$, i.e. the next state is predicted by the dynamics (6) with the disturbance set to zero.

D. Measurement Update

If the constraint (7) from the new measurement agrees with the current state estimate, i.e. if the trajectory from the time update is still feasible, it is also still optimal. Otherwise, the estimator must add the most probable correction to the trajectory so as to satisfy (7), over the entire horizon. This correction will move the estimated position the shortest distance necessary, i.e. to the constraint.

The most influential measurements will be at *events*, when the quantized position measurement changes value. The position is then known to be exactly half way between the two quantization levels some time during the short period between the two measurements.

V. Simplified Event-Based Estimator

Although the JMAP estimation problem is tractable, it is far too demanding to solve in real time on a small micro-controller. In this section, we derive a realistic estimator by simplifying the JMAP estimator. The key simplifications are:

- In the JMAP approach, u_{bias} would be a state variable. We assume that it varies slowly, and can be estimated separately from p and v.
- Active position measurement constraints are considered only at the current time and last event, which is considered as a known position at a known time.
- When applying measurement updates, it is preferred to change the state estimate as little as necessary. This rule is needed since the simplified estimation problem would otherwise be underdetermined.

The states of the estimator will be estimates of the current state $\hat{p}$, $\hat{v}$, and $\hat{u}_{\text{bias}}$ together with a history described by the time of the last event.

A. Time Update

As in the JMAP case, the time update is simply the dynamics of the process model without noise

$$\begin{pmatrix}\hat{p}(k+1)\\ \hat{v}(k+1)\end{pmatrix} = \underbrace{\begin{pmatrix}1 & h\\ 0 & 1\end{pmatrix}}_{A}\begin{pmatrix}\hat{p}(k)\\ \hat{v}(k)\end{pmatrix} + \underbrace{\begin{pmatrix}\frac{1}{2}h^2\\ h\end{pmatrix}}_{B}\Big(u(k) + \hat{u}_{\text{bias}}(k)\Big). \tag{9}$$

There is no systematic drift of u_{bias} in the model, so $\hat{u}_{\text{bias}}$ is not changed in the time update. There is no reason to believe that u_{bias} has changed unless indicated by the measurements.

B. Measurement Update

When the current position measurement y disagrees with the current position estimate $\hat{p}$, i.e. y differs from the quantization of $\hat{p}$ according to (1), a measurement update is applied. The most probable cause of estimation error is taken to be an error Δv in the velocity estimate at the last event—the only error that requires no disturbance after the last event to explain it. By superposition, this leads to a velocity error Δv and a position error $\Delta p = \Delta v \Delta t$ at the current time, where Δt is the time since the last event. Adjusting the position estimate to lie at the constraint $\hat{p} = p$, i.e. to just agree with the measurement y, the measurement update becomes

$$\hat{p}^+ = p = \hat{p} + \Delta p, \qquad \hat{v}^+ = \hat{v} + \frac{\Delta p}{\Delta t}. \tag{10}$$

As in the PI controller with observer, $\hat{u}_{\text{bias}}$ is formed from the time integral of the difference between estimated and measured velocity, i.e., it accumulates the difference between estimated and measured position. The measurement update for $\hat{u}_{\text{bias}}$ thus becomes

$$\hat{u}^+_{\text{bias}} = \hat{u}_{\text{bias}} + \frac{K}{T_i}\Delta p, \tag{11}$$

in analogy to $\hat{U}_{\text{bias}} = \frac{K}{T_i}(Y - \frac{1}{s}\hat{V})$ in (4).

C. Fixes

The gains from position error Δp to velocity and disturbance corrections Δv and Δu_{bias} given above usually work well. Since the estimator is a considerable simplification and because of some unmodeled effects, there is a need to fix some corner cases.

1) LTI Mode Timeout: When there is a long time between events, a position error only indicates a small velocity correction, and the gain from Δp to Δv goes down (as Δt^{-1}). This works as intended for steady motion.

The low gain should no longer be used, however, when the position errors become big, e.g. when the cart is stuck due to friction. The estimator time update will predict a high velocity due to prolonged control signal activity, but the measurement update should actually keep $\hat{v}$ down.

To handle this case, a timeout of $T_{\text{timeout}} = 2T_{\text{filter}}$ is used. If measurement constraints have been active for the last T_{timeout} time, the time constant $\Delta t = T_{\text{filter}}$ of the LTI velocity estimators is used in the measurement update, giving a relatively high, and fixed, gain from Δp to Δv.

2) Measurement Gain Limitation: The case when the time Δt between events is very short is also problematic. Since the gain from errors in Δt and y to Δv becomes very high, any measurement noise that is not captured by the quantization model will be heavily amplified. Since we use the K and T_i parameters from the PI controller, which has been designed to tolerate a lag of T_{filter} in the velocity estimate, we limit Δt in (10) to be no shorter than T_{filter}. Beyond this limit, $\hat{v}$ will behave more like the first order filters of the PI controllers, introducing some low pass filtering on the measurements.

VI. Implementation Issues

In this section, a variety of the issues encountered when implementing the LTI and event-based controllers on a low-end microcontroller are discussed. More details on the implementation can be found in [6].

A. Multi-rate Sampling

The encoder must be sampled every 10 μs = 128 clock cycles to be able to follow cart speeds of up to 5 m/s. There is not much time for calculation in 128 clock cycles and no need to run the controller that often, so it executes at a slower rate. The estimator first updates $\hat{v}$ and $\hat{u}_{\text{bias}}$, and then the controller computes $u = K(\beta v_{\text{ref}} - \hat{v}) - \hat{u}_{\text{bias}}$.

At each invocation of the estimator, it reads the current position. For the event-based estimators, the encoder sampler also saves the direction and time stamp of the last event, allowing the estimator to use a much higher time resolution than its own sampling period. If there was an event during the last sample, the time update is taken up to the event, the measurement update applied, and the time update then taken for the remaining part of the time step. Otherwise, the time update is taken for the whole time step, and the measurement update applied afterwards, if needed.

B. Step by Step Estimator Implementation

The implementation of the event-based estimator requires a number of steps, but with the sequence suggested here, the controller can be verified to work after each. The Basic Event Estimator is an approximation that should be good when events are quite frequent, and is refined gradually. It is naturally implemented starting from an implementation of the Pure PI controller, which supplies the control law that is used throughout:

$$u = K(\beta v_{\text{ref}} - \hat{v}) - \hat{u}_{\text{bias}}$$

(though β is set to one in the end).

1) Basic Event Estimator: The estimates $\hat{p}$ and $\hat{v}$ are constant between events: there is no time update, and measurement updates are at events only. The measurement update (10) is simplified with

$$\hat{v}^+ = \frac{\Delta p}{\Delta t}, \quad \Delta p = \hat{p}^+ - \hat{p},$$

to work when there is no time update for $\hat{p}$. The $\hat{u}_{\text{bias}}$ update mirrors the Pure PI controller's $\hat{U}_{\text{bias}} = \frac{K}{sT_i}(\hat{V} - V_{\text{ref}})$ in (3):

$$\hat{u}_{\text{bias}}(k) = \hat{u}_{\text{bias}}(k-1) + \frac{K}{T_i}\Big(\hat{p}(k) - \hat{p}(k-1) - hv_{\text{ref}}(k)\Big).$$

2) Position Prediction: Now $\hat{v}$ is used to predict the evolution of $\hat{p}$ between events, introducing as time update the relevant part of (9):

$$\hat{p}(k+1) = \hat{p}(k) + h\hat{v}(k).$$

With this time update, the intended form (10) of the measurement update should be used. It is now possible to lower limit Δt by T_{filter} according to section V-C.2; the correction form of the measurement update (10) will ensure that $\hat{v}$ eventually converges to the correct value. The only other visible difference from taking this step is to remove the I-part windup between events when $v \approx v_{\text{ref}}$.

3) Measurement Updates Between Events: Now it is straight forward to apply the measurement update (10) as soon as the position measurement disagrees with $\hat{p}$, allowing the controller to react faster to drops in speed.

4) LTI Mode Timeout: The LTI Mode Timeout fix of section V-C.1 is added. This is needed in the last implementation step to avoid the controller becoming too soft during startups.

5) Full Event Estimator: The full time update (9) is implemented by adding the $\hat{v}$ part, and $\hat{u}_{\text{bias}}$ is now updated only in the measurement update, using (11). Finally, the analysis of the Observer-Based PI controller in section III-B applies, so we set $\beta = 1$ and make T_i four times faster to improve disturbance rejection.

VII. Experimental Comparison

The controllers to be compared are the Pure PI controller, the Observer-Based PI controller, the Basic Event Estimator, the Event Estimator with Position Prediction, and the Full Event Estimator. Since it takes a only minor implementation effort, the Event Estimator with Position Prediction has measurement updates also between events.

To compare the performance of the different controllers under different conditions, a number of step response experiments were performed. In each experiment, the cart begins at rest at position $p = 0$ with all estimates at zero. At time $t = 0$, a step in v_{ref} is made. The cart is allowed to run for 1 m, counting the first 0.3 m as startup and the rest as stationarity. After 1 m, v_{ref} is stepped back to zero.

To have an accurate way to compare the velocity trajectories for the different controllers, the full resolution of the encoder was used for offline evaluation, while in most of the experiments, all feedback was based on a software emulated encoder with the $q = 6$ lowest bits dropped. To reconstruct the velocity trajectory from an experiment, a simple form of the JMAP estimator is used. Only $\hat{p}$ and $\hat{v}$ are used as states, and the effect of u is ignored. Since there is some jitter in the serial communication, the measurement constraints (7) are relaxed to allow that each measurement may have arrived one sample to soon or too late. The optimization problem for the whole trajectory conditioned on all measurements is solved simultaneously.

Fig. 1 shows typical experimental results for the Pure PI and Full Event controllers at $q = 6$, $\omega_c = 20$ rad/s, $v_{\text{ref}} = 0.8$ m/s. We see that the PI controller spends a considerably greater control effort, and that the Full Event controller has slightly better reference tracking.

A. Performance Metrics

To measure the control effort, we use

$$\sigma_u = \frac{\text{std}(u)}{\text{mean}(u)},$$

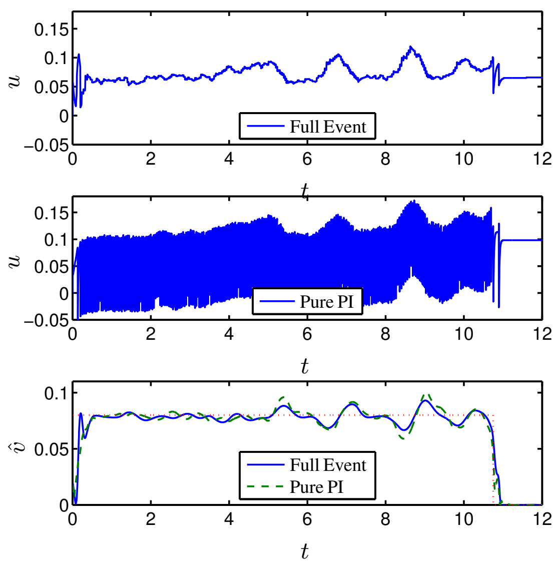

Fig. 1. Experimental results for the Pure PI and Full Event controllers with $\omega_c = 20$ rad/s. The dotted line shows v_{ref}. The friction varies faster at the far end of the track, degrading the velocity tracking in the later half.

where the standard deviation and mean are taken over the last 0.7 m of the trajectory. The mean is taken over all experiments with the same v_{ref}. The reference tracking error is measured by the RMS error over the last 0.7 m, normalized by v_{ref}:

$$E_v = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\frac{\hat{v}(k)}{v_{\text{ref}}} - 1\right)^2}.$$

B. Controller Comparison

To compare the controllers, experiments were made with $v_{\text{ref}} = 0.8$ m and $q = 6$, varying $\omega_c = 5, 10, 20, 40, 80$ rad/s. This gives about 1.7 events/T_{filter} for the Pure PI controller at $\omega_c = 5$ rad/s, and decreasing. For the Full Event and Observer-Based PI controllers, which had reduced T_i, T_i had to be lower bounded to the value used by the Pure PI Controller at $\omega_c = 80$ rad/s to avoid instability.

Fig. 2 compares the control effort and velocity tracking for different controllers and cross-over frequencies. We see that as the control loops become faster, the control effort of the LTI controllers rises much steeper than for the event-based controllers. At first, the velocity tracking improves in much the same way for all controllers, but eventually the Full Event controller wins out.

The greatest gain comes from going from LTI control to the Basic Event controller, but for high bandwidth, there is more to gain with the Full Event controller. The break point where event-based control gives lower control effort than LTI seems to be around one event per T_{filter}, somewhere between $\omega_c = 5$ rad/s and $\omega_c = 10$ rad/s in this experimental setup.

C. Quantization Dependence

To explore quantization effects, the experiments with $\omega_c = 40$ rad/s above were rerun with $v_{\text{ref}} = 0.4$ m/s, varying the quantization as $q = 0 \ldots 6$. The Basic Event Estimator was

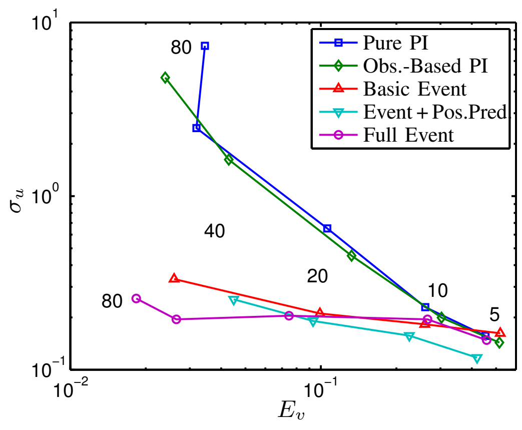

Fig. 2. Experimental comparison of control signal activity σ_u versus velocity RMS error E_v for the LTI controller and three development stages of the event-based controller. Each controller is used with cross-over frequency $\omega_c = 5, 10, 20, 40, 80$ rad/s, (indicated by numbers in the figure) giving successively lower velocity error but higher control signal activity. At $\omega_c = 80$ rad/s, the two simplified event-based controllers become unstable in this case. Each dot corresponds to one experiment.

excluded, since its inability to lower bound Δt made the control signal very noisy at low quantization q.

Figs. 3 and 4 show the tracking error E_v and control effort σ_u with the different encoder resolutions. As the quantization decreases, both E_v and σ_u generally improve, seeming to settle at a quantization free level. The best tracking performance is almost achieved already at $q = 5$, at which point the Full Event controller has also achieved minimum control effort. The Event controller with Position Prediction achieves minimum σ_u at $q = 4$. The control effort of the LTI controllers decreases only gradually, the Observer-Based PI controller being a bit more gentle.

The Full Event controller actually performs at its best with some extra quantization in this case, so there is some room for improvement to make it behave more like the other controllers when events are frequent.

VIII. Conclusion

This paper presented a simple event-based state estimator for a moving cart with quantized position measurements, derived as a simplification of a joint maximum a posteriori (JMAP) estimator. Velocity control based on the event-based estimator was compared experimentally to classical linear time-invariant (LTI) controllers. In the experiments, it was seen that the benefits of event-based control begin to appear when the LTI controllers are unable to filter out the quantization noise efficiently, around the point of one quantization step per velocity filter time constant.

The foremost benefit with event-based control is to greatly reduce the noise in the control signal. The lowered control effort makes it practical to use a much higher gain in the control loop, improving disturbance rejection. Already a simplified event-based controller comes a long way compared to the LTI controllers, but with high controller gain, the full event-based estimator shows superior performance.

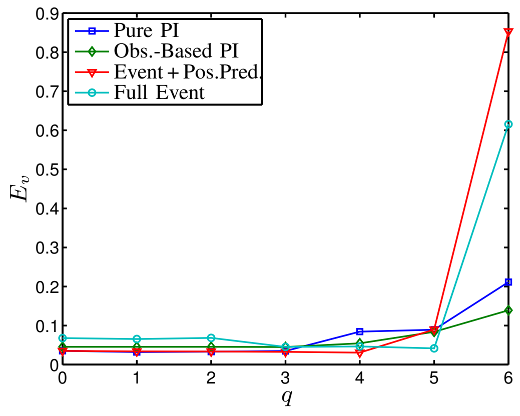

Fig. 3. Experimental comparison of velocity RMS error E_v for LTI and event-based controllers versus number of discarded encoder bits q.

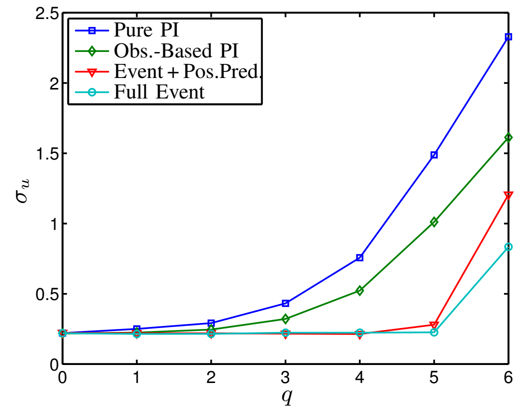

Fig. 4. Experimental comparison of control signal activity σ_u for LTI and event-based controllers versus number of discarded encoder bits q.

Acknowledgments

The authors would like to thank M. Karlsson for helpful comments. This work has been supported by the Swedish Research Council.

References

[1] K.-E. Årzén. A simple event-based PID controller. In *Proc. 14th IFAC World Congress*, 1999.

[2] K. J. Åström and B. Bernhardsson. Comparison of periodic and event based sampling for first-order stochastic systems. In *Proc 14th IFAC World Congress*, 1999.

[3] K. J. Åström and B. Wittenmark. *Computer-Controlled Systems*. Prentice Hall, 3rd edition, 1997.

[4] H. Cox. On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Transactions on Automatic Control*, 9(1):5–12, Jan 1964.

[5] R. E. Curry. *Estimation and Control with Quantized Measurements*. Research Monograph No. 60. M.I.T. Press, 1970.

[6] T. Henningsson. Event-based control and estimation with stochastic disturbances. Licentiate Thesis LUTFD2/TFRT--3244--SE, Department of Automatic Control, Lund University, Sweden, Nov. 2008.

[7] T. Henningsson, E. Johannesson, and A. Cervin. Sporadic event-based control of first-order linear stochastic systems. *Automatica*, 44(11):2890–2895, Nov. 2008.

[8] J. Rawlings and B. Bakshi. Particle filtering and moving horizon estimation. *Computers and Chemical Engineering*, 30(10-12):1529–1541, 2006.

[9] J. H. Sandee, W. P. M. H. Heemels, and P. P. J. van den Bosch. Case studies in event-driven control. In *Proc. Hybrid Systems: Computation and Control*, 2007.

[10] V. Vasyutynskyy and K. Kabitzsch. Simple PID control algorithm adapted to deadband sampling. In *Proc. 12th IEEE Conference on Emerging Technologies and Factory Automation (ETFA'07)*, 2007.