2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID)

2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID) | 979-8-3503-4678-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/VLSID57277.2023.00018

Delay-Aware Control for Autonomous Systems

Sumana Ghosh
Indian Statistical Institute, Kolkata, India, E-mail: sumana@isical.ac.in

***Abstract*—The increasing demand for implementing multiple perception-based real-time applications (e.g., lateral control) onto autonomous systems is introducing new challenges into their control design paradigm, such as the delay-aware design of software controllers for such systems. This becomes more challenging when a variable delay occurs due to the variation of the workload of computation associated with the perception tasks, such as compute-intensive image processing tasks. For such systems, the traditional controller-design-followed-by-implementation paradigm fails to provide the desired control performance all the time. One such example is designing controllers by considering worst-case perception computation delay and thus providing sub-optimal control performance. The primary focus of this work is to address this issue by introducing a novel multi-rate control synthesis approach for autonomous systems. The idea is to apply multiple controllers and adaptively switch among them based on the current delay value without compromising the stability while improving the control performance. We report promising results w.r.t state-of-the-art approaches toward enhancing control performance and optimizing control cost.**

***Index Terms*—Perception-based Systems, Autonomous Systems, Controller Synthesis, Variable Delay, Control Performance**

I. Introduction

Perception-based computing is becoming an integral and important part of autonomous systems nowadays with the growing efficiency of image-processing devices and low cost CMOS cameras having good resolution [11]. Example of such systems ranges from autonomous vehicles to robots to industrial automation systems. Today's autonomous vehicles are equipped with various sensors such as multiple cameras, LiDARS, radars, etc. To know the necessary information about the environment such as relative position, distance, perception/tracking of objects, data from these devices are processed in real time (i.e., *sensing and perception computation task*). Next, this sensed information is fed into the underlying controllers to determine appropriate control inputs such as vehicle speed, acceleration, driving direction, etc (i.e., *control computation task*). Finally, control inputs are actuated to the physical systems via actuators (i.e., *actuation task*).

Large and variable delay in perception computation: The sensing and perception computation task is generally highly compute-intensive because of the involvement of complex and heavy deep learning algorithms (e.g., for object detection). As a consequence, this results in a long sensing delay in the control computation phase. The execution times of perception computation tasks for an image stream depend on image workload variations which inherently depend on the current environmental situation and the image content. For example, an image of a vehicle passing through a congested area is heavy in nature and it needs to process input camera frames at a higher rate to get the desired accuracy, thus leading to a higher computation time. This primarily results in a wide range between the best-case and the worst-case perception computation times, leading to a variable and large sensing delay before the control computation task.

Major issue in controller design phase: The conventional design approach maintains a clear delineation between the task of designing the controllers and the task of implementing the corresponding software control tasks on the embedded processor. Control engineers first choose appropriate sampling periods and then design controllers following a standard control-theoretic practice without bothering with the software implementation architecture details. Instead, they communicate with embedded systems engineers via well-defined interfaces specifying sampling periods and sense-to-actuation delays. Similarly, in the implementation phase, system engineers implement the controllers as software tasks in the embedded computing platform without knowing the details of the controller design steps. However, in the case of designing controllers for autonomous systems, this design-followed-by-implementation paradigm does not work properly because such systems suffer from variable sensing delay due to the huge perception-based computation assigned just before the control computation task. In order to efficiently design such systems,*one should consider the delay variations during the design of the controller*. One existing solution to this problem is to use linear quadratic regulator (LQR) control considering the worst-case workload situation [9]. However, this leads to a pessimistic solution with poor effective resource utilization, and sub-optimal control performance [6] for those systems.

Existing techniques: There is very little research on the impact of perception computing on control performance. Some recent research efforts in this direction are reported in [4, 6–8, 11]. In [4], the authors address the issue of affecting control performance in autonomous systems caused because of the different choices of perception computing tasks and their scheduling decisions. However, they do not consider the delay-sensitivity issues from a controller design perspective as we do in this work. Design of switched control systems and Markov jump linear systems considering workload variations and platform implementation constraints are reported in [6] and [7] respectively. A survey of the recent developments in the field of autonomous vehicle software system and several design challenges in this direction of research are reported in [8]. Though in [2], a system-scenario-based design for embedded systems is introduced, it does not consider the control performance and long delay effect. Instead, this work bridges the design gap by relating delay variation due to perception computation to variable sample rates to control execution sequence and then design multi-rate control scheme for performance enhancement.

Novelty of the proposed work: In this work, we facilitate the usage of multiple controllers for compensating the performance degradation caused due to variable delay. Instead of pessimistically considering the worst-case delay, we examine the actual delay value at each time point and based on that we apply an appropriate controller for the plant from the set of *pre-designed* controllers. Assume that the base controller is designed with a sampling period h selected by consider-

2380-6923/23/$31.00 ©2023 IEEE
DOI 10.1109/VLSID57277.2023.00018

ing the ideal operating conditions with a minimal sense-to-actuation delay. Now, along with this controller, we pre-design $(m-1)$ others controllers considering different possible sense-to-actuation delays, i.e., $2h, 3h, \cdots, mh$. The value of m can be determined from the worst-case delay scenario. We show that application of such multiple controllers outperforms the state-of-the-art three other control design techniques [6, 9].

II. Modeling Delayed Control Systems

1. Plant and Controller Model: The plant model is considered as linear time-invariant systems with the dynamics,

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) \tag{1}$$

where, $x(t) \in \mathbb{R}^n$ defines the plant state, $y(t) \in \mathbb{R}^p$ defines the plant output and $u(t) \in \mathbb{R}^m$ is the control input at time t respectively. The transition, input, and output matrices are described respectively by A, B and C. The sensors measure the plant states periodically at discrete-time points $t_k = kh,\ k = 0, 1, \ldots$. The interval $(t_{k+1} - t_k)$ is known as the sampling period h. Given a sampling period h, the dynamics of the discrete-time system becomes,

$$x[k+1] = \Phi(h)x[k] + \Theta_0(h,\tau)u[k] + \Theta_1(h,\tau)u[k-1]$$
$$y[k] = Cx[k]$$

where, the discrete-time parameters are related to the continuous-time parameters in the following way.

$$\Phi(h) = e^{Ah},\ \Theta_0(h,\tau) = \int_0^{h-\tau} e^{As}Bds,\ \Theta_1(h,\tau) = \int_{h-\tau}^{h} e^{As}Bds.$$

Here, the delay $\tau \leq h$ models the sense-to-actuation delay. Generally, it consists of 1) sensor to processor delay (includes perception computation), 2) control computation time, and 3) processor to actuator delay. Ideally, the delay caused during control computation and actuation is negligible in comparison with the sensing and perception computing delay for an autonomous system.

The discrete-time dynamics of the stabilizing feedback controller is $u[k] = -Kx[k]$ where K is the feedback gain. We assume $u[0] = 0$. By augmenting the system state space $X[k]$ with the previous control input, $u[k-1]$, i.e., $X[k] = \begin{bmatrix} x[k] \\ u[k-1] \end{bmatrix}$, we get the plant dynamics as,

$$X[k+1] = A_d(h,\tau)X[k] + B_d(h,\tau)u[k], \quad y[k] = C_d(h)X[k]$$

where,

$$A_d(h,\tau) = \begin{bmatrix} \Phi(h,\tau) & \Theta_1(h,\tau) \\ 0 & 0 \end{bmatrix}, \quad B_d(h,\tau) = \begin{bmatrix} \Theta_0(h,\tau) \\ 1 \end{bmatrix},$$
$$C_d(h) = [C(h) \quad 0].$$

For the augmented system, we set $\hat{K} = [-K\ \ 0]$ so that $u[k] = \hat{K}X[k]$. The gain K can be computed using any standard techniques such as linear quadratic regulator (LQR), pole-placement, etc. [5], based on $A_d(h,\tau)$ and $B_d(h,\tau)$. Now, the dynamics of the closed-loop system becomes,

$$X[k+1] = A_1(h,\tau)X[k]$$
$$\text{where, } A_1(h,\tau) = \begin{bmatrix} \Phi(h) - \Theta_0(h,\tau)K & \Theta_1(h,\tau) \\ -K & 0 \end{bmatrix}. \tag{2}$$

Note that here the dynamics of the closed-loop system is designed considering one-sample delay.

2. Stability and Control Performance: A system is said to be *asymptotic stable*, if all the eigenvalues of the closed-loop dynamic matrix A_1 lie within a unit circle. The control performance criterion quantifies how fast the system output (y) meets the reference (r) value or how minimal control effort (u) is needed. We consider *settling time* as the control performance metric in this work. It is the time that y takes to reach the reference r and stay within a threshold (e.g., 2 %) of r forever without external disturbances. Note, shorter the settling time better the control performance. To measure the control effort we use the standard *quadratic cost function* of the LQR design technique. For a discrete-time system, it is $J = \sum_{k=0}^{\infty}(x^{\mathsf{T}}[k]\Gamma_1 x[k] + u^{\mathsf{T}}[k]\Gamma_2 u[k])$, where, $\Gamma_1 \succcurlyeq 0$ and $\Gamma_2 \succ 0$ are symmetric matrices that represent the relative weights to the state trajectories and control effort respectively.

3. Drop Scenario due to Delay: Consider the case when $\tau = m \times h$ where $m > 1$, that means, sense-to-actuation delay is more than one sample or multiple of it. In such cases, due to the lack of fresh control inputs at those samples, old control inputs will be applied. We call those samples as *dropped samples*. E.g., suppose due to the delay a dropped sample occurs at the $(k+1)$-th sampling interval, i.e., $[k, k+1)$. Therefore, in this case, the control update equation will be $u[k+1] = u[k]$, i.e., at the k-th and $(k+1)$-th sample, the control input to the plant will remain the same. In such a dropped scenario where the control inputs do not change but the plant continues to evolve can be modeled using the closed-loop dynamics matrix,

$$A_0(h,\tau) = \begin{bmatrix} \Phi(h,\tau) - \Theta_0(h,\tau)K & \Theta_1(h,\tau) \\ O & I \end{bmatrix}. \tag{3}$$

For notational convenience, we will refer to $A_1(h,\tau)$ and $A_0(h,\tau)$ as simply A_1 and A_0 respectively.

4. Control Execution Sequence: For a given plant-controller pair together with its dynamic matrices $\{A_0, A_1\}$, a *control execution sequence* is an infinite computation schedule of the controller, generated by an *infinitely repeating finite length* pattern $\sigma \in \{A_0, A_1\}^*$. Following a given σ having the length l, the dynamics of the closed-loop system becomes,

$$X[k+1] = \sigma[k\%l]X[k], \forall k \in \mathbb{N}.$$

As an example, consider the control execution sequence $\sigma = A_0A_1A_1A_0A_1A_1$. For this, the closed-loop system evolves as,

$$X[6] = A_1X[5] = A_1A_1X[4] = \ldots = A_1A_1A_0A_1A_1A_0X[0].$$

For notational convenience, we represent control execution sequence as binary patterns. E.g., $A_0A_1A_1A_0A_1A_1$ is denoted by $\sigma = 011011$ where '1' represents a control execution/actuation instance, while '0' represents a dropped sample.

III. A Motivational Example

To illustrate that how multiple controllers can mitigate the effect of variable delay in control performance, we take the example of a cruise control (CC) system where the vehicle should maintain a reference speed all the time. The controller monitors the current speed (y) of the vehicle from the sensors, maintains the reference speed level (r) by updating the throttle angle which is the control input u. The dynamics of the linearized third order CC system is adapted from [3]. We set the reference speed to $r = 30\,\mathrm{km/h}$ and

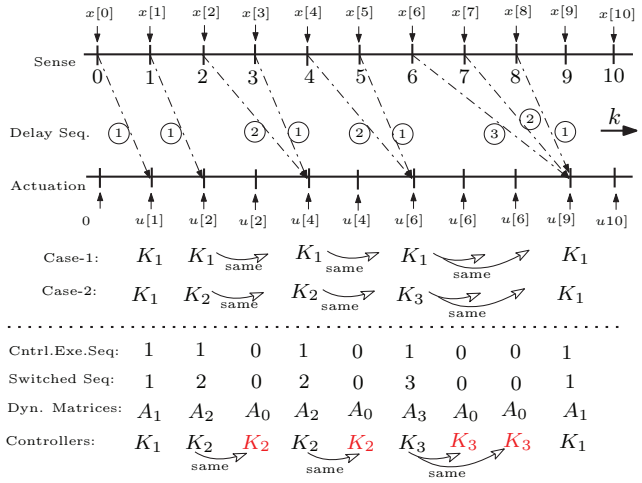


Fig. 1: State evaluation under a delay sequence.

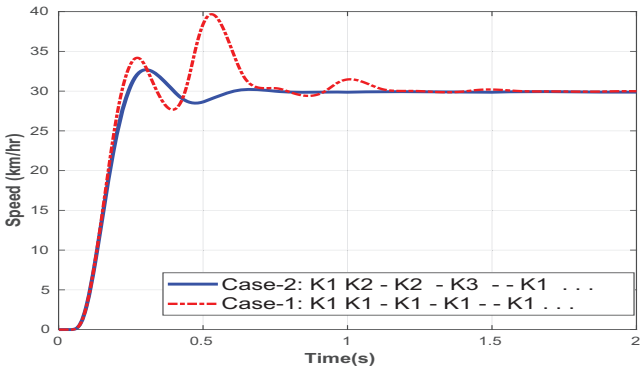


Fig. 2: Output response of the CC system.

the settling time to 1 s. We assume that the sense-to-actuation delay varies within the range [10 ms, 30 ms]. Let D be one such specific sense-to-actuation delay sequence where $D = (10\,\text{ms}, 10\,\text{ms}, 20\,\text{ms}, 10\,\text{ms}, 20\,\text{ms}, 10\,\text{ms}, 30\,\text{ms}, 20\,\text{ms}, 10\,\text{ms})$ Initially, the controller is designed with the *base sampling period* of $h = 10\,\text{ms}$ following the LQR control technique. The respective control gain is $K_1 = [1443.1 \quad 178.3 \quad 11.1]$. We set the initial vehicle speed to $0\,\text{km/h}$. We conduct an experiment of 2 s.

The sense-actuation events of the CC system under this delay sequence D are shown in Figure 1. In the figure, D has been represented in terms of the number of samples (circled one). Essentially, a delay of 10 ms can be written as a 1-sample delay. Similarly, we can write a 2-sample delay for a delay of 20 ms. In this way, D is represented as $\{1, 1, 2, 1, 2, 1, 3, 2, 1\}$ in Figure 1. We assume this delay sequence is repeating in the entire experiment session. Since, at $k = 2$ (i.e., the 2-nd time step) the delay is of 2-sample, at $k = 3$ the old control input is actuated, i.e., $u[3] = u[2]$ due to the lack of fresh control input as shown Figure 1 (see "Actuation" line). Similarly, $u[5] = u[4]$ for a 2-sample delay at $k = 4$ and $u[8] = u[7] = u[6]$ for a 3-sample delay at $k = 6$.

The idea of delay-aware control synthesis emphasizes the fact of adaptive usage of multiple appropriately designed controllers instead of using a single controller on facing variable delay. Figure 2 reflects the advantage of this idea by comparing the output responses of the CC system under the delay sequence D when 1) single controller is used all the time, 2) multiple controllers are used. For Case-1, since a single controller is used, the control input is computed using K_1 only as highlighted in Figure 1. On the other hand, for Case-2, we consider two more controllers having the gain as $K_2 = [668.03 \quad 103.96 \quad 8.25]$ and $K_3 = [313.87 \quad 61.72 \quad 6.28]$ along with K_1, and control input is computed using K_1, K_2, and K_3 following the controller dynamics stated in Section II-1. *The idea is to compensate the performance degradation by applying a different controller in the execution instances immediately preceding some dropped samples caused due to variable delay.* Therefore, as shown in Figure 1, K_2 is used to compute the control input at a time step when there is a 2-sample delay occurs at that time step. Similarly, K_3 is used to compensate the delay-effect for a delay of 3-sample. Note that the closed-loop system has a switching behaviour as: $K_1 \rightarrow K_2 \rightarrow K_3 \rightarrow K_1 \cdots$. Figure 2 shows the improvement in the settling time by approximately 58% for Case-2. It is worth to observe that these multiple controllers are applied respecting the sampling period of $h = 10\,\text{ms}$, hence, leading to the similar configuration of *zero-delay at actuation points*.

IV. Proposed Control Scheme

The the proposed delay-aware multi-rate control scheme is as follows. The first step towards this is to obtain a model of delay characterization so that we can analyze runtime behavior and efficiently design appropriate controllers.

A. Delay Characterization and Lookup Table for Controllers

The delay variation can be easily characterized from the frequently occurring workload scenarios and can be bounded by a minimum (d_{min}) and a maximum (d_{max}) value. Generally, workload scenarios are statistically analyzed and frequently occurring workload scenarios are classified using various mathematical models such as PERT distribution [1]. Note, the exact delay sequence at runtime cannot be determined and can only be assumed to be arbitrary satisfying the predetermined bounds d_{min} and d_{max}. However, with these bounds of delay variation, we can analyze the system behavior for *each* delay value $d \in [d_{min}, d_{max}]$ and precompute suitable controller with gain K_d to take care of the resultant performance degradation. Let the value d reflects a delay of q-samples where $q = \lceil \frac{d}{h} \rceil$ and h be the base sampling period. We then design K_d with the sampling period of qh.

For a closed-loop system, there exists an upper bound M such that a stabilizable controller can be designed for each sampling period in $\{h, 2h, \ldots, Mh\}$. Following the approach presented in [10], we design the *optimal stable* controller with gain K_q for each such sampling period $qh, q \in \{1, 2, \ldots, M\}$ using LQR technique and compute the respective closed-loop dynamic matrix A_q. Next, we store all these controllers and dynamic matrices in a look-up table. In runtime, based on the exact delay sequence respective optimal stable controllers from this look-up table will be selected, and switching among these controllers will be in effect. This step is discussed in the next subsections.

B. Find Control Execution Sequence from Delay Sequence

Let $D = \{d_1, d_2, d_3, \cdots\}$ be a delay sequence that occurs at runtime where $\forall k, \quad d_k \in [d_{min}, d_{max}]$.

Note, d_k denotes the delay amount occurred at the k-th sampling interval for $k \geq 0$. Let h be the base sampling period. Corresponding to D, we can generate a sequence of dropped samples $D_S = \{q_1, q_2, q_3, \cdots\}$ where $q_k = \lceil \frac{d_i}{h} \rceil$. This essentially indicates that a delay amount of d_k is responsible for q_k number of dropped samples. For example, corresponding to $D = \{10\,\text{ms}, 10\,\text{ms}, 20\,\text{ms}, 10\,\text{ms}, 20\,\text{ms}, 10\,\text{ms}, 30\,\text{ms}, 20\,\text{ms}, 10\,\text{ms}, \cdots\}$, we get $D_S = \{1, 1, 2, 1, 2, 1, 3, 2, 1 \cdots\}$ for h=10 ms.

Since q_k denotes the number of dropped samples occurred at the k-th sampling interval, $k + q_k$ essentially indicates the actuation time instance corresponding to the sensing event at the k-th sampling interval. Therefore, for a given $D_S = \{q_1, q_2, q_3, \cdots\}$, we can glean the sequence of actuation time instances as, $D_A = \{a_1, a_2, a_3, \cdots\}$ where $a_k = k + q_k$. For the example of $D_S = \{1, 1, 2, 1, 2, 1, 3, 2, 1 \cdots\}$, we get $D_A = \{1, 2, 4, 4, 6, 6, 9, 9, 9 \cdots\}$ as depicted in Figure 1 (follow the "Actuation" line). Evidently, from the example, all the elements of D_A are not distinct. Let, $\bar{D_A}$ contains only the distinct elements of D_A.

Now, with this $\bar{D_A}$, we can derive the control execution sequence σ where $\sigma[i] = 1$ if $i \in \bar{D_A}$ or 0 otherwise. Note that in σ, '1' denotes a control execution/actuation instance, while '0' denotes a dropped sample (see Section II-4). For $\bar{D_A} = \{1, 2, 4, 6, 9, \cdots\}$, we derive $\sigma = 110101001\ldots$, as illustrated in Figure 1 (see "Cntrl.Exe.Seq" below the dotted line). Recall that at each dropped sample the old control input is actuated (ref. Section II-3). This may cause control performance degradation at those samples due to the lack of fresh control input. To take care of such situation, we allow to switch to a different controller in the lookup table on facing one or more dropped samples in successive sampling intervals.

C. Idea Behind the Multi-rate Control

The reason behind modifying the controller preceding one or more dropped samples is as follows. Consider the control execution sequence $\sigma = 100$ having one control execution instance followed by two consecutive dropped samples. Initially, we have a controller designed with a sampling period h. As discussed earlier, a dropped sample excludes control inputs from being computed freshly, whereas at those dropped samples, the plant evolves following its dynamical equations. Now, according to $\sigma = 100$, while the controller executes once, the plant evolves continuously over a window of $3h$ instead of h. In such a situation, on its execution, the controller does not process the intended plant output, i.e., one sampled with a period of h, but process a plant output sampled with a period of $3h$. Hence, for this instance of controller execution, we use a separate controller which is designed using the period of $3h$ instead of the controller designed using the period of h . Predominantly, for a sequence of the form 10^{q-1}, we apply a new delay compensating controller with a sampling period of qh. Next section presents how to get those sampling periods $\{qh\}$ from a control execution sequence.

D. Control Execution Sequence to Set of Sampling Periods

Before discussing how to get those sampling periods, we first describe some important terminologies.

Definition 1 (Drop Subsequence). *A drop subsequence of an l-length control execution sequence $\sigma[1 \ldots l]$ is a q-length substring of σ, with the form 10^{q-1}, for some $q \in \{1, 2, \ldots, l\}$. That means, it starts with a 1 followed by $(q-1)$ consecutive 0s. When σ is recurrent, if $\sigma[j] = 1$ for some j, $1 \leq j \leq l$, the $(q-1)$ consecutive 0s are at position $\sigma[(j+1)\%l], \sigma[(j+2)\%l], \ldots, \sigma[(j+q-1)\%l]$ respectively.* □

For example, in the control execution sequence $\sigma = 0110101110$, we get three distinct drop subsequences, 1, 10, and 100 with $q = 1, 2, 3$ respectively. In case of 100, the leading 1 is at the position $\sigma[9]$, whereas the two successive 0s are at $\sigma[10]$ and $\sigma[1]$ respectively, capturing the recurrent behavior of the sequence. In contrast, there are two occurrences of 10, respectively at the positions $(\sigma[3], \sigma[4])$ and $(\sigma[5], \sigma[6])$.

Note that for a drop subsequence, 10^{q-1}, the corresponding sequence of system dynamic matrices representing the system evolution over q consecutive sampling intervals is $A_1 A_0^{q-1}$ (ref. Section II-4). For compensating the drops, we replace the subsequence $A_1 A_0^{q-1}$ by $A_q A_0^{q-1}$ where A_q is the closed-loop dynamic matrix computed considering the control gain K_q executed in the *same execution slot* as the A_1 at the start of the subsequence 10^{q-1}. This leads to the *switched control execution sequence* which is formally defined as:

Definition 2 (Switched Control Execution Sequence). *Given a control execution sequence $\sigma[1, .., l]$ over $\Sigma = \{0, 1\}$, the switched control execution sequence $\hat{\sigma}[1, .., l]$ defined over $\Sigma = \{0, 1, 2, ..., q\}$ is obtained by substituting each drop subsequence 10^{q-1} of σ, with the subsequence $q0^{q-1}$.* □

For example, corresponding to the control execution sequence $\sigma = 110101001$, we get $\hat{\sigma} = 120203001$ by replacing each occurrence of drop subsequences 10 and 100 of σ with 20 and 300 respectively (see "Switched Seq." in Figure 1).

We refer to the controller K_1 corresponding to A_1, as the *base controller*. Note that the compensating controller K_q corresponding to A_q is designed for the sub-sequence $q0^{q-1}$. As discussed in Section IV-A, K_q is already designed with the sampling period qh and stored in the lookup table. In other words, K_1 is designed assuming that the control actuation will be there at a regular interval of h, whereas, K_q is designed with a priori information that there is no control actuation to be taken place within the next qh time unit, hence, *K_q is more delay-aware for a delay of q-samples*. Moreover, as illustrated in Figure 1 and discussed in Section III and earlier in this section, K_q will be applied at the *same* time point as the K_1, therefore, it ensures a zero gap in consecutive control actuation instance and consequently leads to a better control performance.

It is worth to mention that replacing 10^{q-1} with $q0^{q-1}$ simply replaces the control algorithm executed in the same execution slot, and thus the utilization of processor-bandwidth remains the same. Therefore, starting from an initial state $X[0]$, following $\hat{\sigma}[1, 2 \cdots l] = q_1 0^{q_1 - 1} \cdots q_\Delta 0^{q_\Delta - 1}$ with total Δ drop subsequences such that $l = q_1 + q_2 + \ldots + q_\Delta$, the switched dynamical system evolves as:

$$\begin{aligned} X[l] =& A_0^{q_\Delta - 1} A_{q_\Delta} \cdots A_0^{q_j - 1} A_{q_j} \ldots A_0^{q_1 - 1} A_{q_1} X[0] \\ =& A_{\hat{\sigma}[l]} A_{\hat{\sigma}[l-1]} \cdots A_{\hat{\sigma}[2]} A_{\hat{\sigma}[1]} X[0]. \end{aligned}$$

For notational convenience, in the above equation, we use $A_{\hat{\sigma}[j]}$ for denoting the product matrix $A_0^{q_j - 1} A_{q_j}$. Note that within total M drop subsequences, if m subsequences are

distinct, then the scheduler switches among m different controllers. As an example, for $\hat{\sigma} = 1202013001$, we use three distinct controllers K_1, K_2, and K_3 for the sampling periods $h, 2h$, and $3h$ respectively. The corresponding closed-loop dynamic matrices A_1, A_2, and A_3 are assigned at the head of each occurrence of the three distinct drop subsequences 1, 20, and 300 respectively. The pictorial illustration of this is given in Figure 1 (see "Dyn. Matrices" and "Controllers").

Sequence Extracted Sampling Periods: For each switched control execution sequence $\hat{\sigma}[1, \cdots, l]$, we have the set of *sequence extracted sampling periods* as $H_{\hat{\sigma}} = \{q_1 h, \ldots, q_M h\}$ corresponding to the M drop subsequences of distinct lengths. For example, given the $\hat{\sigma} = 1202013001$, we get $H_{\hat{\sigma}} = \{h, 2h, 3h\}$ corresponding to its three distinct drop subsequences, namely, 1, 20, and 300. Next, for each of these M sequence extracted sampling periods in $H_{\hat{\sigma}}$, we select the respective optimal stable controller from the look-up table (ref. Section IV-A) and use in runtime. In this way, switching among multiple multi-rate controllers takes place on facing a delay sequence to avoid performance degradation.

E. Stability during Switching

In case of a switched dynamical system, the entire switched system can be unstable for some switching sequences, though the system is stable in all individual states (note, each state respects a choice of the sampling period and a stable controller is there at that state) [5, 10]. In order to get rid of such a situation, it is necessary to ensure the stability of switched systems under arbitrary switching [5]. This can be achieved by showing the existence of a Common Quadratic Lyapunov Function (CQLF) for all the states. There exists several well-defined methods for finding CQLF [5]. In this work, we follow the method provided in [10] to design optimal stable controllers for different sampling periods, and thus ensure system stability against any arbitrary switching between themselves.

V. Case Study and Results

To demonstrate the effectiveness of the proposed approach, in this section, we illustrate our results on a case study of vision-based lateral control (LC) system [6]. In an LC system, the vehicle has to follow a lane autonomously. The perception computation is done to process the camera frames and compute the lateral deviation at a set look-ahead distance. The controller computes the control input which is the front wheel steering angle after receiving the lateral deviation as the system output from the sensors. The fifth-order LC system has the following dynamics.

$$A = \begin{bmatrix} -10.06 & -12.99 & 0 & 0 & 0 \\ 1.096 & -11.27 & 0 & 0 & 0 \\ -1.0 & -15 & 0 & 15 & 0 \\ 0 & -1 & 0 & 0 & 15 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 75.47 \\ 50.14 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$C = [0 \;\; 0 \;\; 1 \;\; 0 \;\; 0]$.

The five states variables of the LC system are 1) lateral velocity, 2) yaw rate of the vehicle, 3) lateral deviation from the desired centerline at the look-ahead distance, 4) the angle between the tangent to the road and vehicle orientation, and 5) the curvature of the road at the look-ahead distance.

Initially, we design the *base controller* with gain K_1 for the sampling period of $10\,\mathrm{ms}$ using the LQR technique. At first, we set the system output to $0\,\mathrm{m}$. Now, it is the control objective to return the output to the reference values of $0.03\,\mathrm{m}$ and $-0.03\,\mathrm{m}$ as shown in Figure 4. We conduct the overall experiment for a time duration of $6\,\mathrm{s}$ on MATLAB version R2022b. We assume that the sense-to-actuation delay can vary within the range $[10\,\mathrm{ms}, 30\,\mathrm{ms}]$. For the demonstration purpose, we consider the same recurring delay sequence $D = (10\,\mathrm{ms}, 10\,\mathrm{ms}, 20\,\mathrm{ms}, 10\,\mathrm{ms}, 20\,\mathrm{ms}, 10\,\mathrm{ms}, 30\,\mathrm{ms}, 20\,\mathrm{ms}, 10\,\mathrm{ms})$ as mentioned in Section III, however, the method works for any given D occurred at runtime.

Due to the maximum delay of $30\,\mathrm{ms}$, apart from K_1, we design two other LQR controllers with gains K_2 and K_3 for the sampling periods of $20\,\mathrm{ms}$ and $30\,\mathrm{ms}$ respectively. We compare the proposed method denoted as **MULTI** with other three control design schemes as discussed next.

WC: This is the *worst-case delay oriented control scheme* [9]. Here, the sampling period of the LC system is chosen as $30\,\mathrm{ms}$ since it is the maximum delay in our setting. We design an LQR controller for $30\,\mathrm{ms}$. This approach generally suffers from a lower control performance because of the higher sampling period. However, it exhibits a better control effort due to less aggressive control gains.

SLC: This is the *switched linear control design scheme* [6]. Here, LQR controllers are designed for the sampling periods of $10\,\mathrm{ms}$, $20\,\mathrm{ms}$, and $30\,\mathrm{ms}$. Note that the main difference between SLC and MULTI is twofold; 1) In MULTI, we allow switching among controllers K_1, K_2, and K_3, respecting the same base sampling period of $10\,\mathrm{ms}$. Whereas SLC allows switching among the sampling periods, i.e., $10\,\mathrm{ms}$, $20\,\mathrm{ms}$, and $30\,\mathrm{ms}$, which in turn leads to the switching among K_1, K_2, and K_3. 2) In MULTI, we ensure periodic actuation (i.e., zero delay between two consecutive actuation instances) since we do not allow to change the base sampling period of $10\,\mathrm{ms}$, and thereby, we use the old control input at each dropped sample caused due to variable delay. Contrastly, in SLC, with the varying delay since sampling periods get changed, it causes aperiodic actuation sequences. A pictorial illustration of the difference in control actuation instances together with the choice of controllers for these two methods is shown in Figure 3. The dotted arrow indicates the previous control input is applied at that point.

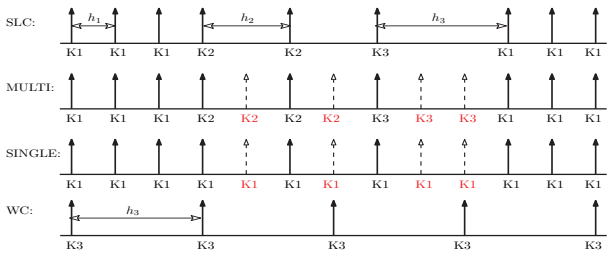


Fig. 3: Difference in control actuation instances.

SINGLE: In the *single-rate control scheme*, the sampling period is chosen as $10\,\mathrm{ms}$, and an LQR controller is designed. SINGLE differs from MULTI by applying the same controller all the time, as elaborately discussed in Section III. It suffers from both the lower control performance and higher control cost since it applies the highly aggressive controller K_1 all the time, as illustrated in Figure 3.

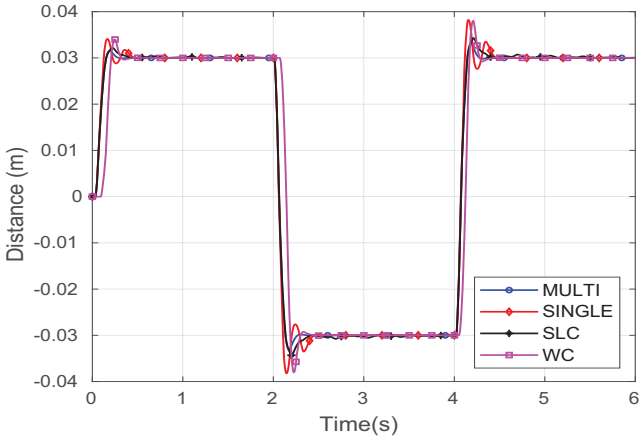
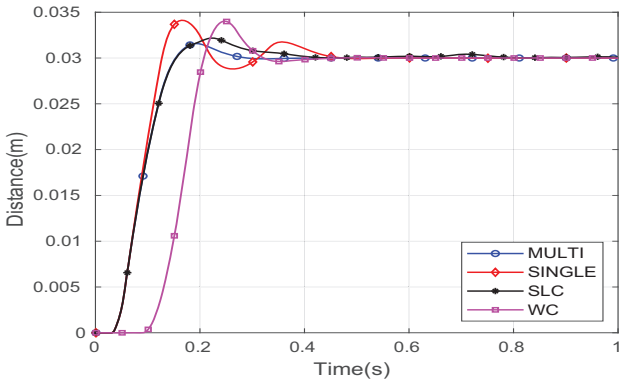
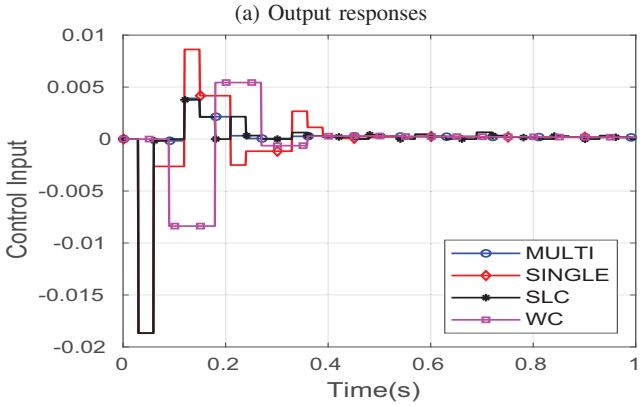


Fig. 4: Output response of LC under different schemes.

Figure 4 compares the output responses of the LC system obtained by applying the proposed method MULTI and these three existing control design schemes. It is clear from Figure 4 that the system settles down quickly with a settling time of $0.35\,\mathrm{s}$ when the proposed approach is applied, as compared to other control schemes. This can be further cleared from Figure 5a that highlights the output response over a time window of first $2\,\mathrm{s}$. In particular, our proposed approach achieves



(a) Output responses



(b) Control efforts

Fig. 5: Improvement in control performance & control cost.

approximate improvements of $32\,\%$, $27\,\%$, and $22\,\%$ in settling time compared to SINGLE, WC, and SLC respectively.

A comparison of control efforts taken by these four methods is depicted in Figure 5b. Note, control performance and control cost are considered as contradictory in nature. Better performance can be achieved at the cost of higher control effort only, hence, optimizing both together is very difficult. Towards this, MULTI exhibits a more balanced results in terms of both the control performance and control cost. Evidently from Figure 5b, with MULTI comparatively lower control cost is achieved as compared to SLC and SINGLE. This is because, less aggressive but optimally designed LQR controllers K_2 and K_3 are used judiciously with the aggressive controller K_1 on facing the variable delay. MULTI shows better control performance (i.e., lesser setting time) since it ensures periodic actuation with the period h $=10\,\mathrm{ms}$, and at each actuation instant, the appropriate control input is applied exploiting suitable control gains that are calculated particularly for compensating performance degradation caused due to the variable delay. On the other hand, in SLC, because of aperiodic actuation, lower control performance is achieved w.r.t MULTI. With SINGLE both the settling time and control cost suffer from poor performance, whereas WC shows better control cost but poor control performance. Over the entire simulation of $6\,\mathrm{s}$, the cumulative LQR control cost achieved with these four methods are: 1.7915 (MULTI), 1.8217 (SLC), 1.8356 (SINGLE), 0.5834 (WC) - which exactly resembles our discussion.

VI. Concluding Remarks

Feedback control loops implement many essential features (e.g., safety-critical functionalities) of today's autonomous systems. Since most of these features need some perception computation to process heavy sensory data before the control computation step, the controller synthesis process suffers from variable delay which affects the control performance of such systems. This work presents a potential solution to this issue by developing a delay-aware controller design scheme. The proposed scheme reports significantly improved results w.r.t other three existing control schemes. Currently, the proposed method considers a single-task setting – this can be extended to a multiple-task setting as an immediate next step.

References

[1] S. Adyanthaya et al. Robustness analysis of multiprocessor schedules. In *Proc. SAMOS*, pages 9–17, 2014.
[2] S. V. Gheorghita et al. System-scenario-based design of dynamic embedded systems. *ACM Trans. Des. Autom. Electron. Syst.*, 14(1), jan 2009.
[3] S. Ghosh et al. A structured methodology for pattern based adaptive scheduling in embedded control. *ACM Transactions on Embedded Computing Systems*, 16(5s):189:1–189:22, 2017.
[4] C. Hobbs et al. Perception computing-aware controller synthesis for autonomous systems. In *Proc. DATE*, pages 457–462, 2021.
[5] D. Liberzon. *Switching in systems and control*. Springer, 2003.
[6] S. Mohamed et al. Optimising quality-of-control for data-intensive multiprocessor image-based control systems considering workload variations. In *Proc. Euromicro DSD*, pages 320–327, 2018.
[7] S. Mohamed et al. Designing image-based control systems considering workload variations. In *Proc. CDC*, pages 3997–4004, 2019.
[8] S. D. Pendleton et al. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1), 2017.
[9] S. Saidi et al. Special session: Future automotive systems design: Research challenges and opportunities. In *Proc. CODES+ISSS*, pages 1–7, 2018.
[10] M. Schinkel, Wen-Hua Chen, and A. Rantzer. Optimal control for systems with varying sampling rate. In *Proc. ACC*, 2002.
[11] E. P. van Horssen et al. Switched lqg control for linear systems with multiple sensing methods. *Automatica*, 103:217–229, 2019.