Implementation-aware design of image-based control with on-line measurable variable-delay

Róbinson Medina*‡, Sander Stuijk*, Dip Goswami*, Twan Basten*†
*Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
†ESI, TNO, Eindhoven, The Netherlands
‡TNO Powertrains Department, Helmond, The Netherlands

***Abstract*—Image-based control uses image-processing algorithms to acquire sensing information. The sensing delay associated with the image-processing algorithm is typically platform-dependent and time-varying. Modern embedded platforms allow to characterize the sensing delay at design-time obtaining a delay histogram, and at run-time measuring its precise value. We exploit this knowledge to design variable-delay controllers. This design also takes into account the resource configuration of the image processing algorithm: sequential (with one processing resource) or pipelined (with multiprocessing capabilities). Since the control performance strongly depends on the model quality, we present a simulation benchmark that uses the model uncertainty and the delay histogram to obtain bounds on control performance. Our benchmark is used to select a variable-delay controller and a resource configuration that outperform a constant worst-case delay controller.**

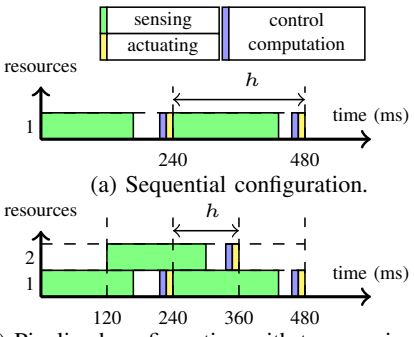
I. Introduction and Related Work

Image-Based Control (IBC) is becoming common in Cyber-Physical Systems (CPS) because of the possibility of using cameras and image-processing algorithms as sensors. Image processing algorithms are capable of sensing objects for which no simple sensor exists, at a cost of additional sensing delay and a platform with additional processing resources. Examples of IBC are found in ADAS (Advanced Driver Assistance Systems) [1] and visual servo control [2].

An IBC is commonly implemented using a sequential resource configuration (see Fig. 1a). This implementation considers the actuation period of the system to be longer than the sensing delay. However, long sensing delays result in long actuation periods which potentially limits control performance (i.e. the Quality of Control (QoC)) [3]. The QoC can be improved using a controller with pipelined resource configuration [4]–[6] (see Fig. 1b). A pipelined controller uses multiprocessors with sufficient processing resources to compute the sensing algorithm in a pipelined fashion (assuming independence between subsequent samples). This increases the actuation rate, which improves QoC.

These control strategies use the constant worst-case execution time of the sensing delay in the image-processing algorithm in their designs [4]–[6]. However, for image-processing algorithms, the most common delays are grouped in one or more modes (corresponding e.g. to the number of objects or regions of interest being processed) while worst-case delay is unlikely to happen (see Fig. 2) [7]. A worst-case design is therefore conservative. An alternative design philosophy is to consider the variable delay in the controller design.

This work was funded by the Dutch NWO Domain Applied and Engineering Sciences (TTW) as part of the Robust Cyber-Physical Systems (rCPS) program, project 12697.



(a) Sequential configuration.

(b) Pipelined configuration with two sensing cores.

Fig. 1: Examples of resource configurations of IBC using the worst-case latency of the image-processing algorithm. A shorter sampling period h is achieved by pipelined control.

The design of variable-delay controllers has been widely studied in the literature (e.g. [8]–[10]). These approaches mainly focus on analysing the controller robustness under the assumptions that the delay varies in a bounded range and it cannot be measured on-line (i.e. the delay value is not available before the controller is computed, e.g. because of a transfer of sensed data over a wireless network). Although these are valid assumptions in many applications, in IBC the delay can be characterized off-line with a histogram of delays. Additionally, once the image-processing algorithm has finished its execution, the delay can be on-line measured using existing timing mechanisms (e.g. built-in timers) [11]. This information can be made available to the controller to improve its QoC.

We propose an alternative to the existing variable-delay control design strategies. Our approach designs a set of controllers based on implementation-dependent histograms of delays, and on-line measurements of the delay. We compensate for the variable delay with a modification of the predictor approach

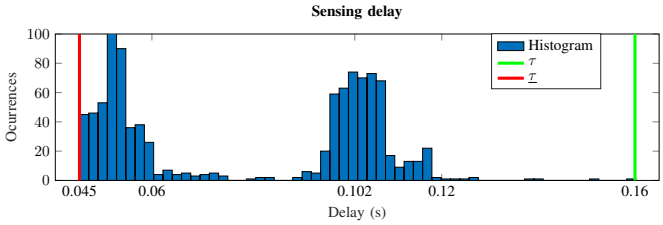


Fig. 2: Example sensing delay histogram in an experiment with 10000 executions of an image-processing algorithm.

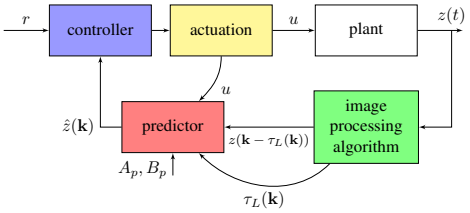


Fig. 3: Variable-delay predictor controller.

proposed in [10]. This predictor estimates the states using model-based predictions with a constant delay. The proposed predictor in this work estimates states using a variable delay. Ideally, our estimator cancels the effect of the delay resulting in a QoC similar to the delay-free case. However, in practice prediction errors occur due to model uncertainties. Therefore, we introduce a simulation benchmark that obtains QoC bounds based on model uncertainties. The benchmark is used to select a resource configuration and one controller such that the QoC is improved compared to a constant worst-case design.

Contributions: we propose an implementation-aware variable-delay control design. The novelty of the design lies in the combination of implementation-dependent information (i.e. histogram of delays, measured delay, and resource configurations) with model-based predictions to compensate for the variable delay while improving QoC. We also provide a simulation benchmark that uses model uncertainties and delay histogram to select a controller and a resource configuration that improves QoC.

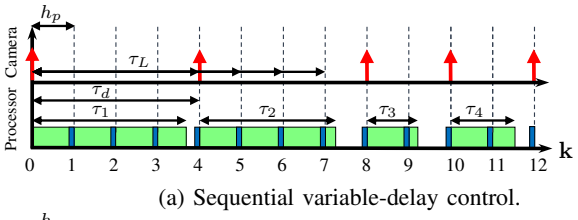
II. Variable-delay predictor controller

A variable-delay predictor controller has the structure presented in Fig. 3. The variable delay is compensated by the prediction block, which estimates the current system states. The controller uses this estimation to compute new inputs.

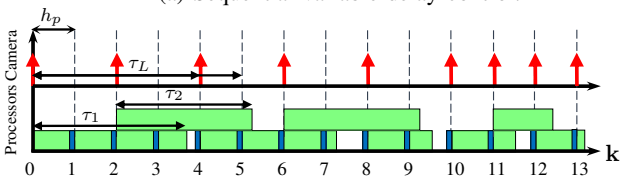
A. Run-time behaviour

An example of the resource usage of a predictor controller is shown in Fig. 4. The controller is composed of a sensing task (i.e. an image-processing algorithm), a prediction task, a control computation task, and an actuation task. The actuation period h_p denotes the interval between two consecutive controller updates. The prediction, control computation, and actuation are periodic tasks with period h_p. This timing is assumed strictly constant. In case the sensing task is still running when the periodic tasks need to start (e.g. near the end of time instant $\mathbf{k} = 1$), the sensing task is temporarily pre-empted. The sensing task starts its execution when the previous sensing task is finished and at a time multiple of h_p (i.e. $\mathbf{k}h_p$). The sensing task is therefore non-periodic.

The use of pipelined sensing control creates communication overhead when sharing information between processing resources. For example, in Fig. 4b the upper pipe sends the output of the image-processing algorithm to the lower pipe which computes the periodic tasks. Typically the size of this information is limited to a few floating point numbers (e.g. one floating point number per measured output in our case study). This overhead is negligible compared to the image-processing algorithm delay. Additionally, note that at time



(a) Sequential variable-delay control.



(b) Pipelined variable-delay control with two sensing cores.

Fig. 4: Red arrows represent an image capture. Blue blocks represent the prediction, control computation and actuation tasks. The sensing delay varies with the sequence $\tau(t) = \{\tau_1, \tau_2, \tau_3, \tau_4, \ldots\}$. The prediction length $\tau_L(\mathbf{k})$ is updated from the end of τ_1 until a new computation is finished.

$\mathbf{k} = 10$ in the pipelined case of Fig. 4b, both processors finish their computation in the same time slot. The prediction then uses the output based on the most recent image.

B. System modelling

Given the continuous-time system:

$$\dot{x}(t) = A_c x(t) + B_c u(t),\ y(t) = Cx(t - \tau(t)) \qquad (1)$$

with $x(t) \in \mathbb{R}^n$ the states, $u(t) \in \mathbb{R}^i$ the control input. $A_c \in \mathbb{R}^{n\times n}$, $B_c \in \mathbb{R}^{n\times i}$, and $C \in \mathbb{R}^{o\times n}$ are the state, input, and output matrices respectively. We assume that A_c is a stable matrix. $\tau(t)$ is the time-varying sensing delay bounded by $\underline{\tau} \leq \tau(t) \leq \tau$, with $\underline{\tau}$ and τ the best and worst-case delay respectively.

The discrete-time version of Eq. 1 is given by:

$$z(\mathbf{k}+1) = A_p z(\mathbf{k}) + B_p u(\mathbf{k}),\ y(\mathbf{k}) = Cx(\mathbf{k} - \tau_d(\mathbf{k})) \quad (2)$$

with $A_p \in \mathbb{R}^{n\times n}$ and $B_p \in \mathbb{R}^{n\times i}$ the discrete-time prediction state and input matrices discretized with the actuation period h_p. $\tau_d(\mathbf{k}) \in \mathbb{Z}^+$ is the discretized variable delay given by $\tau_d(\mathbf{k}) = \lceil \frac{\tau(t)}{h_p} \rceil$ [12].

C. Controller block

Real-time controllers commonly use time-domain metrics (e.g. settling time, overshoot, etc.) as QoC measurements because of their impact on system-level performance. Consequently the QoC in our application is defined as $QoC = S_t^{-1}$, where S_t is the controller settling time. The controller block is based on the assumption that the predicted state is equal to the current system state, i.e. $\hat{z}(\mathbf{k}) = z(\mathbf{k})$. From the control perspective the sensing delay is therefore cancelled by the predictor. The controller is then designed for the system of Eq. 2 without taking into account the effect of the delay.

D. Predictor block

The predictor block (see Fig. 3) estimates the system states at time $\mathbf{k}$. The predictor uses Proposition 1 to estimate the states. The predictions are based on the available information: the delayed state measurement $z(\mathbf{k}-\tau_L(\mathbf{k}))$, past controller inputs $u(\mathbf{k}-\tau_L(\mathbf{k})+j)$ (with $0 \leq j \leq \tau_L(\mathbf{k})-1$), the discrete-time model of Eq. 2, and the prediction length $\tau_L(\mathbf{k})$. $\tau_L(\mathbf{k})$ is the age of the measurement, as explained in Section II-E. Our predictor is inspired by the one presented in [10], which considers $\tau_L(\mathbf{k})$ constant (and un-measurable) in the predictions. Our predictor uses on-line measurements of the delay to compensate for the variable delay, which creates a prediction with variable number of actuation periods (i.e. in Proposition 1 below, $\tau_L(\mathbf{k})$ is updated every time a prediction is needed). This cancels the effect of the sensing delay guaranteeing controller stability.

Proposition 1. *Predictor with variable prediction length: given the discrete-time system of Eq. 2 with sensing data measured $\tau_L(\mathbf{k})$ actuation periods ago, if the states of the plant at time $\mathbf{k}$ are estimated using:*

$$\hat{z}(\mathbf{k}) = A_p^{\tau_L(\mathbf{k})} z(\mathbf{k}-\tau_L(\mathbf{k})) + \sum_{j=0}^{\tau_L(\mathbf{k})-1} A_p^{\tau_L(\mathbf{k})-1-j} B_p u(\mathbf{k}-\tau_L(\mathbf{k})+j)$$

then $\hat{z}(\mathbf{k}) = z(\mathbf{k})$.

Proof. The proof follows the ideas of [10]. □

E. Prediction length

Because the sensing delay is typically longer than the actuation period, there are cases when the predictor does not have new sensing information to compute the states. Therefore, the same sensing information is used more than once to estimate the current state. For example, in both resource configurations of Fig. 4 the first discrete-time delay τ_1 corresponds to $\tau_d(\mathbf{k}) = 4$ actuation periods. This means that a sensing task is completed after time $\mathbf{k} = 3$ and before $\mathbf{k} = 4$, while the image was captured at $\mathbf{k} = 0$; therefore the prediction length at $\mathbf{k} = 4$ is $\tau_L(\mathbf{k}) = 4$ actuation periods. In the next time step, no new sensing information is available. Therefore, $\tau_L(\mathbf{k}) = 5$ actuation periods. The prediction length is summarized in Table I for both configurations. In the sequential case, the prediction length varies from $\tau_L(\mathbf{k}) = 2$ to $\tau_L(\mathbf{k}) = 7$ actuation periods, while for the pipelined case it ranges from $\tau_L(\mathbf{k}) = 2$ to $\tau_L(\mathbf{k}) = 5$ actuation periods.

The prediction length bounds for a $\tau_d(\mathbf{k})$ are defined as:

$$\underline{\tau_L(\mathbf{k})} = \tau_d(\mathbf{k}) = \left\lceil \frac{\tau(t)}{h_p} \right\rceil, \overline{\tau_L(\mathbf{k})} = \tau_d(\mathbf{k}) + \left\lceil \frac{\tau_d(\mathbf{k})}{\gamma} \right\rceil - 1 \quad (3)$$

with $\underline{\tau_L(\mathbf{k})}$, $\overline{\tau_L(\mathbf{k})} \in \mathbb{Z}^+$ the lower and upper bound of the prediction length in actuation periods respectively, and $\gamma \in \mathbb{Z}^+$ the number of processing resources. In the sequential case $\gamma = 1$ while in the pipelined case $\gamma > 1$. Adding more sensing resources (i.e. increasing γ) only affects $\overline{\tau_L(\mathbf{k})}$ because pipelining the processing algorithm produces the same sensing delay while more sensing information is generated in between consecutive sensing tasks. Note that controllers with different h_p can have the same $\overline{\tau_L(\mathbf{k})}$. However, such controllers still predict over a different range of time. To compare the upper bound of the prediction length of controllers with different actuation periods, we use:

$$\overline{\tau_L(t)} = \overline{\tau_L(\mathbf{k})} * h_p \quad (4)$$

with $\overline{\tau_L(t)} \in \mathbb{R}^+$ the prediction length upper bound in seconds.

TABLE I: Prediction length in terms of actuation periods in the configurations of Fig. 4

current time $\mathbf{k}$	last sensed information time		prediction length $\tau_L(\mathbf{k})$	
	sequential	pipelined	sequential	pipelined
1	X	X	X	X
2	X	X	X	X
3	X	X	X	X
4	0	0	4	4
5	0	0	5	5
6	0	2	6	4
7	0	2	7	5
8	4	4	4	4
9	4	4	5	5
10	8	6,8	2	2
11	8	6,8	3	3
12	10	10	2	2
13	10	11	3	2

F. Prediction quality

In an ideal case, the model-based predictions perfectly estimate the system states after the variable delay. However, in practice it is common to have a margin of error in the system model (e.g. model uncertainties) which affects the quality of the predictions and therefore the QoC. The prediction length plays also a role in the prediction quality: long predictions rely more on the model and control inputs than on the measurements. Proposition 1 shows that with a long prediction (i.e. $\tau_L(\mathbf{k}) \to \infty$) the measurement term $A_p^{\tau_L(\mathbf{k})} z(\mathbf{k}-\tau_L(\mathbf{k})) \to 0$ because the spectral radius of A_p satisfies $\rho(A_p) < 1$, while the second term has always several larger terms given by the sum of elements $A_p^{\tau_L(\mathbf{k})-1-j} B_p u(\mathbf{k}-\tau_L(\mathbf{k})+j)$. Model uncertainties have therefore a strong negative impact in long predictions, much stronger than in short predictions.

G. QoC benchmarking model

For the QoC benchmarking of the variable-delay controller, we include uncertainties in the dynamic model:

$$\dot{x}(t) = (A_c + \Delta A_c)x(t) + (B_c + \Delta B_c)u(t) \quad (5)$$

with ΔA_c and ΔB_c the uncertain part of the state and input matrices respectively. The uncertain matrices are obtained from the tolerance of the elements A_c and B_c.

III. Variable-delay control design

The procedure for designing our variable-delay controller with sequential or pipelined resource configuration is the same. This procedure is detailed in the next subsections.

A. Histogram characterization

We select a set of actuation periods that potentially improve QoC. An actuation period similar to the worst-case delay, i.e. $h_p \rightarrow \tau$ produces a QoC similar to a worst-case based design. The benefit of using our controller appears when the actuation period is chosen significantly smaller than the worst-case delay. Ideally, a short actuation period $h_p \rightarrow 0$ is preferred because the resulting QoC is similar to the continuous-time controller with no delay. However, a small actuation period causes a long $\tau_L(\mathbf{k})$, which is strongly affected by model uncertainties (see Section II-F). The controller actuation periods are therefore chosen taking into account the modes in the delay histogram. To do so, we apply the following steps:

1. Identification of modes in delay histogram: we define a mode as a peak or a group of close peaks in the delay histogram [7]. The centre points of the modes are grouped as $\mathbb{T} = \{\tau_1, \tau_2, \ldots, \tau_m\}$. We only consider modes that are significantly shorter than the worst-case delay because a sampling period selected based on a mode near the worst case produces little or no improvement on QoC. E.g., in Fig. 2 $\mathbb{T} = \{51, 102\}$ (all times in ms).

2. Selection of representative delays: a set of Representative Delays $\mathbb{D} = \{D_1, D_2, \ldots, D_d\}$ is selected, such that each element of the set covers (i.e. it is larger than) at least one mode. If two modes are close to each other, they are combined in one representative delay. E.g., in Fig. 2 $\mathbb{D} = \{60, 120\}\ ms$.

These two steps can be automated using the scenario identification technique presented in [7].

B. Selection of actuation periods and resource configuration

A set of actuation periods $\mathbb{H} = \{h_{p1}, h_{p2}, \ldots, h_{ph}\}$ and a resource configuration γ are defined according to the desired QoC and robustness. We follow these steps:

1. Actuation period selection: we select $\mathbb{H}$ to guarantee QoC improvement when $\mathbb{D}$ occurs in the platform. The actuation periods are defined as $\mathbb{H} = \frac{\mathbb{D}}{\underline{\tau_L}}$ with $\underline{\tau_L}$ the desired minimum prediction length. $\underline{\tau_L}$ is a design parameter chosen based on the desired QoC and the model reliability. For example, a large $\underline{\tau_L}$ strongly enhances the QoC when a highly reliable model (i.e. a model with low uncertainty) is available; a small $\underline{\tau_L}$ can still enhance the QoC when a less reliable model is available. More than one value of $\underline{\tau_L}$ can be considered for each actuation period. Continuing the running example, with 10% uncertainty in A_c we choose $\underline{\tau_L} = 2$, resulting in $\mathbb{H} = \{30, 60\}$.

2. Resource configuration selection: we select γ to guarantee control robustness. Therefore, we limit the prediction length $\overline{\tau_L(t)}$ to a desired value. To do so, for each actuation period in the set $\mathbb{H}$ we use Eqs. 3 and 4 with $\tau(t) = D_d$ to select a γ such that $\overline{\tau_L(t)}$ stays within a desired bound. $\overline{\tau_L(t)}$ is used because it allows to compare multiple actuation periods. D_d is used since it gives the longest $\overline{\tau_L}$. A long $\overline{\tau_L}$ produces predictions that strongly rely on the system model generating a less robust controller (see Section II-F). A short $\overline{\tau_L}$ produces the opposite effect. If $\overline{\tau_L}$ is longer than desired while insufficient processing resources are available, a shorter $\underline{\tau_L}$ has to be selected in the previous step. Continuing the example with $h_p = 30\ ms$ and $D_2 = 120\ ms$, we aim to have $\overline{\tau_L(t)} \leq 250\ ms$. Using $\gamma = 1$ in Eqs. 3 and 4 gives $\overline{\tau_L(\mathbf{k})} = 7$ and $\overline{\tau_L(t)} = 210\ ms$, meeting the design requirement.

C. Control design

One controller is designed per element in $\mathbb{H}$. To do so, we select a control law with integral action:

$$u(\mathbf{k}) = K\hat{z}(\mathbf{k}) + K_i \int (C\hat{z}(\mathbf{k}) - r) \tag{6}$$

where K is the feedback gain, K_i is the integral gain, and r is the reference. The integral action is necessary for correcting steady-state errors produced by model uncertainties. To design K and K_i in Eq. 6, we use the well-known Linear Quadratic Regulator (LQR) controller that includes integral action [13]. LQR is used because it provides a gain that is optimal to a cost function and it is easily adaptable to MIMO problems. However, other control strategies (e.g. pole placement) could have been used to determine the control gains. LQR trades off control effort with state deviation by minimizing the following cost function:

$$J = \min \sum_{\mathbf{k}=0}^{\infty} z_i^T(\mathbf{k}) H z_i(\mathbf{k}) + u^T(\mathbf{k}) R u(\mathbf{k}) \tag{7}$$

where z_i are the system states augmented with the integral states, $H \in \mathbb{R}^{(n+o)\times(n+o)}$ and $R \in \mathbb{R}^{i\times i}$ are the state and input weight matrices of the LQR, which are tuning parameters. Finding the correct values of H and R is a hard task, because they do not have a direct equivalence to the resulting settling time. Therefore, we use Particle Swarm Optimization (PSO) to find H and R that give a controller with the minimum settling time [6].

D. QoC benchmarking to select best controller

Simulations are used to benchmark the QoC. Each controller is tested using the uncertain model of Eq. 5. Note that Proposition 1 and the control design of Section III-C do not consider model uncertainties. Therefore, simulations are used to test the controller stability. One simulation is run with each delay in $\mathbb{D}$. The delay is kept constant during the entire simulation. This procedure gives bounds on the QoC: the best case QoC is expected with D_1 (i.e. the smallest element in $\mathbb{D}$) while the worst-case QoC is expected with D_d (i.e. the largest element in $\mathbb{D}$). In a run-time simulation, the QoC typically varies between the QoC obtained with the benchmarked delays, because those are the most common delays occurring in the platform.

To compare if there is any QoC gain, a controller with a constant worst-case delay h_{wc} is designed for optimized QoC and tested with the uncertain model. The QoC of both controllers is compared. We select the actuation periods which show QoC improvement compared to the worst-case design. If no improvement is achieved, the procedure of Section III-B has to be repeated with a smaller $\underline{\tau_L}$ or with a larger γ.

IV. Case study: xCPS

eXplore Cyber-Physical Systems (xCPS) is an assembly line that puts together circular complementary blocks [14]. xCPS is equipped with conveyor belts that transport the blocks through multiple actuators to complete the assembly process. The block speed needs to be regulated in the assembly process. A camera and an image-processing algorithm are used as sensor to detect block speed. An IBC can then regulate the block speed by changing the input to the belts. A better QoC (i.e. shorter settling time) means that the blocks move faster between actuators. Consequently, they are assembled faster, which potentially improves the productivity of xCPS.

System model: the conveyor belt model is given by:

$$A_c = \begin{bmatrix} -41.59 & 27.62 \\ -1 & 0 \end{bmatrix}, B_c = \begin{bmatrix} 0.69 \\ 1 \end{bmatrix}$$

where the system states are $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} \omega & \int (u-\omega)dt \end{bmatrix}^T$, ω is the block angular speed. u is the belt input.

We consider that the values on the state matrix can vary up to 15% of their nominal value. No uncertainty is considered in the input matrix. Therefore, Eq. 5 is given by:

$$\Delta A_c = \begin{bmatrix} -6.23 & 4.14 \\ -0.15 & 0 \end{bmatrix}, \Delta B_c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad (8)$$

Image-processing algorithm: the image-processing algorithm on xCPS uses Hough-transform for circles to detect the blocks speed ω. We consider the delay histogram of Fig. 2 for this example. In this histogram, the two modes are produced by the number of blocks being detected in the image: the first mode corresponds to one block, while the second mode corresponds to two blocks. The delay bounds are $\underline{\tau} = 45\ ms$ and $\tau = 160\ ms$. τ is significantly larger than the modes. A variable-delay controller can potentially improve the QoC compared to a worst-case design. Two processing resources are available for executing the Hough-transform and the IBC.

V. Simulation results

We use the procedure of Section III to design variable-delay controllers with sequential and pipelined configurations.

A. Histogram characterization

Using the procedure of Section III-A and Fig. 2, we identify $\mathbb{T} = \{51, 102\}$. Next, we select $\mathbb{D} = \{60, 120\}$.

B. Selection of actuation periods and resource configuration

Using the procedure of Section III-B and the uncertainty described in Eq. 8, we choose to have $\underline{\tau_L(\mathbf{k})} = \{1, 2, 4\}$, resulting in $\mathbb{H} = \{15, 30, 60, 120\}$. These actuation periods are smaller than a worst-case sampling period $h_{wc} = 160\ ms$. Therefore, QoC improvement is possible.

To select the resource configuration, we set $\overline{\tau_L(t)} \leq \underline{180\ ms}$. Considering $\gamma = 1$, Eqs. 3 and 4 result in $\overline{\tau_L(t)} = \{225, 210, 180, 120\}$ per element of $\mathbb{H}$. Only $h_p \in \underline{\{60, 120\}}$ meet the prediction constraint. Considering $\gamma = 2$, $\overline{\tau_L(t)} = \{165, 150, 120\}$ for $h_p \in \{15, 30, 60\}$. In this case all actuation periods meet the minimum robustness constraint. $h_p = 120\ ms$ is not considered because that actuation period is larger than a worst-case actuation period (with $\gamma = 2$, $h_{wc} = 80\ ms$). $\gamma = 2$ is then the preferred resource configuration. However, we evaluate both configurations with all the actuation periods for demonstration purposes.

C. Control design

Using the control design strategy of Section III-C, one controller per element of $\mathbb{H}$ is designed. Likewise, a constant worst-case control is designed to compare whether there is QoC gain. For the sequential worst-case implementation the actuation period is $h_{wc} = 160\ ms$. For the pipelined case, the worst-case $h_{wc} = 80\ ms$.

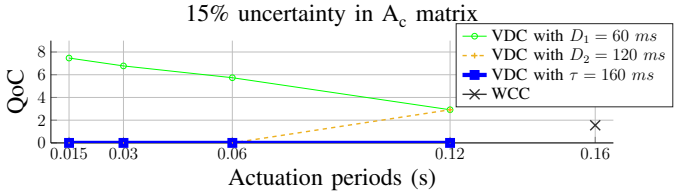
We tried to compare the QoC of our controllers with the predictor approach of [10]. This approach considers τ_L constant in the predictor of Section II-C. The control stability against prediction length errors (i.e. when $\tau_L \neq \tau_L(\mathbf{k})$) must be verified through the Linear Matrix Inequalities (LMIs) of Theorem 1 in [10]. Further, the control stability against model uncertainties and prediction length errors is verified in Theorem 2 in [10]. In our approach, prediction length errors are not present since $\tau_L(\mathbf{k})$ is on-line measured and updated at every prediction; the stability against model uncertainties is addressed using the benchmark presented in Section III-D. To compare QoC of both approaches, we attempted to solve the LMIs of Theorem 1 of [10] for our case study (i.e. our controllers, delay range, and multiple values of τ_L). For the range of delay, Theorem 1 could not find any feasible stable solution for the controller. This is mainly because of the conservative nature of the analysis for dealing with lack of information on the delay behaviour. A quantified comparison of QoC with the two approaches is therefore not possible.

D. QoC benchmarking

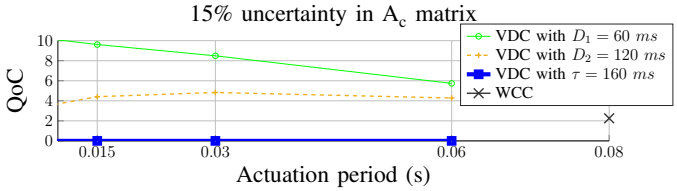
Using the procedure of Section III-D, the QoC of the afore-designed controllers is benchmarked. For demonstration purposes, we also included $\tau = 160\ ms$ in the benchmarked delays. To find the QoC, the settling time is measured when the reference is changed. If the system output does not stabilize within a 2% margin, the QoC is set to zero (i.e. the system output keeps oscillating around the reference).

Fig. 5 shows the benchmark. In both configurations, the best QoC is achieved with $D_1 = 60\ ms$, because it gives the shortest $\tau_L(\mathbf{k})$. Increasing the delay deteriorates the QoC for all h_p. The controllers do not settle when τ is evaluated. However, Fig. 2 shows that the τ is unlikely to continuously occur. Therefore, D_2 is considered as the lower bound on QoC. In practice, the sensing delay will take any value between $\underline{\tau}$ and τ. However, the QoC will be between D_1 and D_2.

With $\gamma = 1$, the QoC is 0 when $h_p \in \{15, 30, 60\}$ and $D_2 = 0.12\ ms$. Note that $h_p \in \{15, 30\}$ does not meet the robustness constraint while $h_p = 60\ ms$ barely meets it. This implies that the controller cannot tolerate the uncertainty. These actuation periods are therefore not desirable in a final implementation. $h_p = 120\ ms$ meets the robustness constraint and shows a QoC improvement with D_2 compared to the worst-case design. $h_p = 120\ ms$ is preferable in a final implementation. With $\gamma = 2$, the same actuation periods produce a better QoC than with $\gamma = 1$. Additionally, in the pipelined case all the controllers



(a) Sequential variable-delay control.



(b) Pipelined variable-delay control.

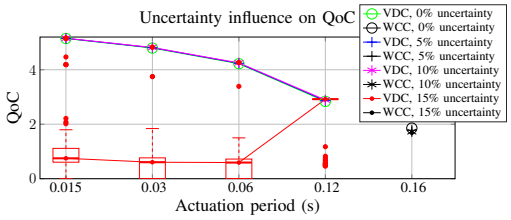
Fig. 5: QoC benchmarking of Variable-Delay Controllers (VDC) and the Worst-Case design Control (WCC) using constant delays D_1, D_2 and τ.

show improvement compared to the worst-case design. The recommended actuation period for $\gamma = 2$ is $h_p = 15\ ms$ because it shows the best QoC improvement.

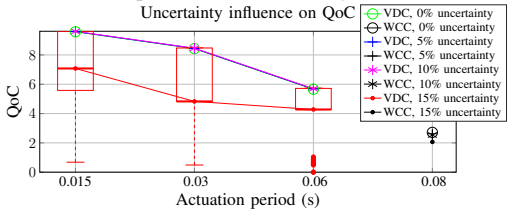
Fig. 6 shows an average QoC of 100 simulations using the controllers, uncertainty in the A_c matrix, and varying $\tau(t)$ generated according to Fig. 2. For the 15% uncertainty we use a Box type of plot. The boxes correspond to 25 and 75 percentiles. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using a red dot. Both graphs also show the effect of uncertainties between 0% and 10%. For simplicity, only the median of the 100 simulations is plotted with these uncertainties. In this case, the QoC deterioration is negligible which means that the smallest h_p can be used.

Fig. 6a shows that the resulting median QoC with 15% uncertainty is better than a worst-case design only with $h_p = 120\ ms$. The rest of the median QoC is close to zero, which makes them undesirable in a final implementation.

Fig. 6b shows that the best QoC is achieved with the



(a) Sequential variable-delay control.



(b) Pipelined variable-delay control.

Fig. 6: QoC with 100 different delay sequences. The lines connect the median QoC. A box plot is drawn for the largest uncertainty.

smallest h_p, which outperforms a worst-case design. The median QoC is also between the bounds presented in Fig. 5b. Note that the lower whisker extends to a QoC which is below the worst-case design. This occurs because some delay sequences contained multiple values close to τ which extends the controller settling time.

VI. Conclusions and Future Work

We presented sequential and pipelined variable-delay control design strategies for Image-Based Control (IBC). Both controllers compensate for the variable nature of the delay using a predictor, which estimates the system states using a system model, the delayed state measurement, and online measurements of the sensing delay. We presented a simulation benchmark that uses the implementation-dependent delay histograms and model uncertainties to select a resource configuration and a controller that improves performance compared to a constant worst-case design.

Our results show that in a system without uncertainties, the shortest actuation period yields the best control performance. However, uncertainties have a strong negative impact on quality of the predictions, much stronger with a longer prediction length, which in turn affects the control performance. Therefore, the prediction length is decreased by increasing the actuation period or adding processing resources in a pipelined fashion. Extra sensing resources improve the control performance and the controller robustness.

References

[1] K. Jo *et al.*, "Development of autonomous car—part I: Distributed system architecture and development process," *Industrial Electronics, IEEE Trans. on*, vol. 61, no. 12, pp. 7131–7140, 2014.

[2] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robotics and Automation*, vol. 13, pp. 82–90, 2006.

[3] P. Sharkey and D. Murray, "Delays versus performance of visually guided systems," *Control Theory and Applications, IEE Proc.*, vol. 143, no. 5, pp. 436–447, 1996.

[4] M.-Z. Yuan *et al.*, "Function block-based pipelined controller," in *Proc. IECON*. IEEE, 2003, pp. 1956–1961.

[5] R. Medina *et al.*, "Reconfigurable pipelined sensing for image-based control," in *Proc. SIES*. IEEE, 2016.

[6] R. Medina *et al.*, "Exploring the trade-off between processing resources and settling time in image-based control through LQR tuning," in *Proc. SIGAPP SAC*. ACM, 2017.

[7] S. V. Gheorghita, T. Basten, and H. Corporaal, "Profiling driven scenario detection and prediction for multimedia applications," in *Proc. of SAMOS VI*, 2006, pp. 63–70.

[8] R. Lozano *et al.*, "Robust prediction-based control for unstable delay systems: Application to the yaw control of a mini-helicopter," *Automatica*, vol. 40, no. 4, pp. 603 – 612, 2004.

[9] P. Garcia *et al.*, "Robustness with respect to delay uncertainties of a predictor-observer based discrete-time controller," in *Proc. of the 45th IEEE CDC*, 2006, pp. 199–204.

[10] A. Gonzalez *et al.*, "Robustness of a discrete-time predictor-based controller for time-varying measurement delay," *Control Engineering Practice*, vol. 20, no. 2, pp. 102 – 110, 2012.

[11] ARM, "ARM Cortex-R7 MPCore Technical Reference Manual," ARM, Tech. Rep., 2014.

[12] K. J. Åström and B. Wittenmark, *Computer-controlled Systems (3rd Ed.)*. Prentice-Hall, Inc., 1997.

[13] S. Carrière, S. Caux, and M. Fadel, "Optimal lqi synthesis for speed control of synchronous actuator under load inertia variations," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 5831–5836, 2008.

[14] S. Adyanthaya *et al.*, "xCPS: A Tool to Explore Cyber Physical Systems," *SIGBED Rev.*, vol. 14, no. 1, pp. 81–95, 2017.