Designing a controller with image-based pipelined sensing and additive uncertainties

RÓBINSON MEDINA, JUAN VALENCIA, SANDER STUIJK, and DIP GOSWAMI, Eindhoven University of Technology, The Netherlands
TWAN BASTEN, Eindhoven University of Technology and ESI, TNO, The Netherlands

Pipelined image-based control uses parallel instances of its image-processing algorithm in a pipelined fashion to improve the quality of control. A performance-oriented control design improves the controller settling time with each additional processing resource, which creates a resources-performance trade-off. In real-life applications, it is common to have a continuous-time model with additive uncertainties in one or more parameters that may affect the controller performance and the aforementioned trade-off. We present a robustness analysis framework for performance-oriented pipelined controllers with additive model uncertainties. We present a technique to obtain discrete-time uncertainties based on the continuous-time uncertainties for given uncertainty bounds. To benchmark such uncertainty bounds for a real system, we consider uncertainties in one element of the system, potentially caused by multiple uncertain parameters in the model. Robustness and its impact in the trade-off analysis are studied. We also provide a robustness-oriented pipelined controller design that takes into account the benchmarked uncertainties. Our results show that in performance-oriented designs, the tolerable uncertainties for a pipelined controller decrease when increasing the number of pipes. In robustness-oriented designs, the controller robustness is enhanced with each newly added pipe. We show the feasibility of our technique by implementing a realistic example in a Hardware-In-the-Loop simulation.

CCS Concepts: • **Hardware → Process variations**; **Hardware-software codesign**; Yield and cost optimization; • **Computer systems organization → Multicore architectures**; *Embedded hardware*; *Embedded software*; *Sensors and actuators*;

Additional Key Words and Phrases: Image-based control; Pipelined sensing control; Robustness analysis; Particle Swarm Optimization; Trade-off analysis; LQR tuning.

ACM Reference Format:
Róbinson Medina, Juan Valencia, Sander Stuijk, Dip Goswami, and Twan Basten. 2018. Designing a controller with image-based pipelined sensing and additive uncertainties. *ACM Transactions on Cyber-Physical Systems* 1, 1, Article 1 (June 2018), 25 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Image-processing algorithms are nowadays being used inside control loops as sensors. Examples of these control loops are found in Advanced Driver Assistance Systems (ADAS) [17, 18] or in visual servo control applications [6]. The main advantage of using image-processing in the loop is the capability of generating sensing information that is not easily acquired with regular sensors. For example, in ADAS it allows to determine traffic signals or road conditions, and in visual servo control

This work is funded by the NWO Domain Applied and Engineering Sciences (TTW) as part of the Robust Cyber-Physical Systems (rCPS) program, project 12697.
Authors' addresses: Róbinson Medina; Juan Valencia; Sander Stuijk; Dip Goswami, Eindhoven University of Technology, De Rondom 70, Eindhoven, 5612 AP, The Netherlands; Twan Basten, Eindhoven University of Technology, ESI, TNO, Eindhoven, The Netherlands.

XXXX-XXXX/2018/6-ART1 $15.00
https://doi.org/0000001.0000001

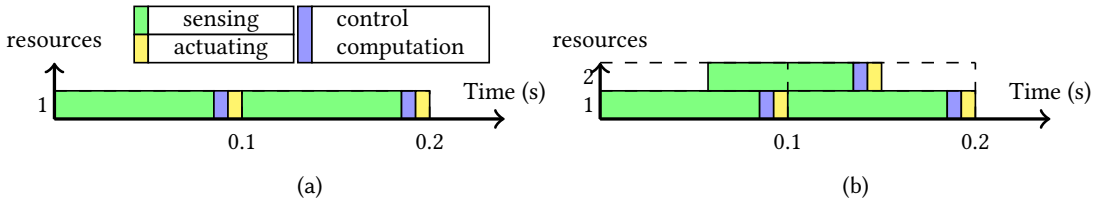

Fig. 1. Example resource configuration of image-based control. a) Sequential configuration. b) Pipelined configuration with two sensing cores.

it allows to acquire the position of particular objects. Image sensors introduce a computational delay that appears as an additional source of latency in the control loop. When latency is longer than the desired sampling period of the system, the lack of sensing information reduces the Quality of Control (QoC) [33].

The rise of hardware with parallel processing capabilities allows to cope with long sensing delays by either parallelizing the image-processing algorithm or by executing it in a pipelined fashion. Parallelization can potentially reduce the sensing latency [2, 19, 36, 44]. However, parallelization is not applicable to all algorithms, it may furthermore be time consuming, and the resulting latency might not be shorter than the desired sampling period of the system. An alternative is to pipeline the sensing algorithm [22, 29, 45]. Pipelining consists of running several instances of the image-processing algorithm in a pipelined fashion. Such a technique generates extra sensing information that decreases the sampling period while the sensing latency remains the same. Pipelining the sensing algorithm can be straightforwardly applied to any image-processing algorithm that does not have any temporal dependencies between subsequent samples (e.g. images). Fig. 1 compares a classical implementation of an image-based controller (i.e. sequential implementation) with a pipelined implementation of two-sensing-cores. Note that the computational delay remains constant because the worst-case execution time is used in the controller design. In the pipelined sensing, doubling the sensing resources doubles the actuation rate of the controller, halves the sampling period, and can be used to improve the QoC. Fig. 2 shows an example of how controllers respond (pipelined and sequential) with the resource configurations of Fig. 1. The settling time (i.e. QoC) of the controller is shortened by using extra sensing information in the control algorithm. The QoC improvement results in performance improvements for the application, which potentially reduces the operation costs. Such a reduction is commonly larger than the cost of the additional processing resources, which motivates the use of multi-core technology.

With performance-oriented designs, pipelining the sensing algorithm potentially improves QoC with each newly added pipe; therefore a trade-off exists between resource usage (i.e. cost of implementation) and QoC. We introduced a method to analyse such a trade-off using Particle Swarm Optimization (PSO) in [30]. The method allows to find the number of sensing cores that still gives a meaningful improvement in QoC. The method assumes perfect knowledge of the system model. However, in practice it is common to have uncertainties in the model. This leads to a robustness constraint. Uncertainty degrades QoC, may cause instability [47, Chapter 8.1], and affects the aforementioned trade-off analysis. For example, in the motivational example of [30], the trade-off analysis indicated that four pipes is the number of cores that still gives a meaningful improvement in QoC. However, if there is a maximum uncertainty of 2% in one of the elements of the state and input matrices (i.e. there is a robustness constraint that the controller continues to be stable with up to 2% deviation in one element of these matrices), the robustness of the controller

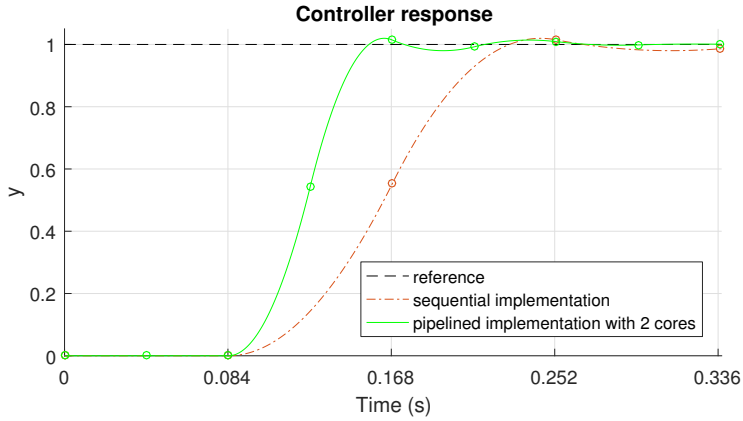

Fig. 2. Example of controller response with and without pipeline. The round markers denote multiples of the sampling period.

with four pipes is lower than a minimum desired bound, whereas the one with three pipes remains within the desired bounds, while it still gives a meaningful improvement in QoC.

There is substantial literature dealing with the stability analysis of a controller based on a discrete-time model with delay and uncertainties (see for example [13, 40]). Additive time-variant norm-bounded uncertainties are common in such analysis because in many physical systems the exact value of the parameters is not known but it is limited to a range of possible values (see for example [4, 5, 24, 34]). Analysing the effect of time-variant bounded uncertainties has not been done yet to pipelined systems. Moreover, discrete-time robustness analysis techniques commonly start from known discrete-time uncertainties (see for example [10]). The relation of such discrete-time uncertainties with real-life continuous-time applications is not obvious. In the trade-off analysis of pipelined systems, this relation is particularly relevant because each newly added pipe produces a different sampling time implying that new discrete-time uncertainties have to be computed. In other words, the same continuous-time uncertainties have different effects in pipelined sensing controllers with different resource configurations. This affects the trade-off analysis.

Contributions: our contributions are threefold: (i) we benchmark discrete-time uncertainties based on continuous-time time-variant norm-bounded uncertainties described by matrices with a single non-zero element (i.e. a single uncertain element in each matrix). The resulting uncertainties are additive, time-variant, and norm-bounded. (ii) We present a robustness analysis technique that shows the impact of the benchmarked uncertainties in the performance-oriented pipelined controllers and in the trade-off between resource usage and QoC. For that we adapt the technique of [13, 40] for pipelined systems. (iii) We present a robustness-oriented controller design that uses the benchmarked uncertainties to enhance robustness with each added pipe. Our contributions capture the interplay between processing resources, control performance, and control robustness in pipelined systems. We show the feasibility of our approach by implementing a pipelined controller on a platform with parallel processing capabilities using Hardware-In-the-Loop simulation.

The paper is organized as follows. Section 2 starts with related work. The state-of-the-art pipelined control strategy is summarized in Section 3. Section 4 shows a design flow that summarizes our approach. The steps of the flow are elaborated in Section 5 with a strategy to structure discrete-time uncertainties in pipelined systems, Section 6 with a technique to benchmark the discrete-time uncertainties based on continuous-time uncertainties, Section 7 with a technique to analyse the

robustness of a performance-oriented pipelined controller and a technique to design robustness-oriented pipelined controllers, and Section 8 with an example analysis. We show the feasibility of our approach with a Hardware-In-the-Loop simulation in Section 9. Section 10 concludes.

2 RELATED WORK

Pipelined sensing control has been used for systems with image-based sensing delay in [7, 22, 23, 29, 30] and for systems with network-based sensing delay in [38, 39, 45]. This literature focuses on comparing pipelined sensing with sequential sensing (e.g. [22]), using frequency domain information to analyse the phase margin of different pipelined controllers [45], allocating resources in networked control to achieve pipeline parallelism e.g. [38], and introducing modelling (e.g. [29]) and control design strategies (e.g. [30]). Analysing the QoC of pipelined control with respect to a model with uncertainties has not been reported before.

The impact of bounded additive uncertainties on control systems has been widely studied in the robust control literature for the continuous-time domain [20, 41, 42] and the discrete-time domain [9, 10, 48]. A pipelined sensing control corresponds to a discrete-time controller; therefore we focus on discrete-time techniques. Discrete-time approaches commonly assume a known set of discrete-time uncertainties. In a pipelined sensing control, the additive uncertainties originate from the dynamic system in the continuous-time domain, whereas its impact has to be analysed on the discrete-time pipelined controller. Therefore, a strategy to discretize a model with uncertainties is required. Strategies to compute discrete-time approximations of the uncertain matrices are found in using the Chebyshev quadrature [34], switched systems [15] or first order forward Euler approximations [21, 24, 37]. However, these strategies correspond to approximations which might not accurately represent the continuous-time system. In pipelined control, each resource configuration varies the sampling period generating different discrete-time uncertainties. The above approximations might not produce enough difference between resource configurations, making them not suitable for this application. Taylor series have been used in [5] to discretize continuous-time uncertainty using the first i terms of the series. The resulting model is a homogeneous polynomial of degree i. Increasing i in the series expansion leads to a higher accuracy in the discretization at the cost of a more complex discrete-time model. This technique might be applicable to pipelined control; however the resulting homogeneous polynomials are not commonly used in discrete-time robustness analysis techniques. We present a technique to benchmark additive bounded discrete-time uncertainties, that eases the robustness analysis.

The combined effect of delay and discrete-time bounded uncertainties on discrete-time controllers has been analysed following two approaches: predictor output [11, 13, 28] or static feedback [13, 40]. Predictor output approaches use an estimate of the system output after the delay to compute the controller output. Static feedback approaches include the delay in the discrete-time model to compute the controller output. The pipelined controller designed in [30] corresponds to a static feedback approach. Static feedback literature with additive uncertainties has reported a control design strategy in the presence of constant and variable time delay in [40] and an uncertainty maximization technique with variable time-delay for a given controller in [13]. In this paper, we apply the static feedback technique to pipelined systems which have discretized time-variant norm-bounded uncertainty. We analyse the effect of model uncertainties on the trade-off between processing resources and QoC in a performance-oriented design, while meeting a robustness constrain. We also develop a robustness-oriented pipelined controller design.

3 PIPELINED CONTROL

This section describes the state-of-the-art pipelined control strategy. To this end, we start with a motivational example of a system that can potentially benefit from pipelined control. Then, the modelling strategy of [29] and the performance-oriented control design strategy of [30] are explained

and applied to this example. If these strategies are applied to multiple resource configurations, the trade-off between the number of pipes and settling time can be analysed.

3.1 Example of pipelined control design

We use as a motivational example, the dynamic model presented in [30]. Consider a control application with two states $x = [x_1\ x_2]^T$, one input $u = u_1$, one output $y = x_1$, and the following matrices:

$$A_c = \begin{bmatrix} 0 & 1.7 \\ -9 & -2.5 \end{bmatrix}, B_c = \begin{bmatrix} 0 \\ 10 \end{bmatrix}, C = [1\ 0] \tag{1}$$

with A_c, B_c, and C_c the nominal state, input, and output matrices respectively. A camera and an image-processing algorithm form a sensor for measuring the states x_1 and x_2. The total sensor-to-actuator delay (using worst-case execution times) is $\tau = 0.084\ s$. The open loop step response has a rise time of $R_t = 0.337\ s$ and a settling time of $S_t = 2.810\ s$. This second order system could represent, for example, the speed change of a motor driving a conveyor belt in an assembly line. The image-processing algorithm could be a Hough transform algorithm which recognizes the objects moving on the conveyor belt.

The sensing configurations shown in Fig. 1 are used to design two controllers with different resource configurations: one with a sequential implementation and one with a two-cores pipelined implementation. Ideally, the sampling period of the system is chosen such that the transient response of the system is sufficiently sampled. For example, the system rise time can be used to approximate the sampling period [31, Chapter 2.9]: $h \approx \frac{R_t}{10}$. However, due to the large τ, the minimum feasible sampling period in a sequential configuration is $h = \tau = 0.084\ s$. With a two sensing cores pipelined configuration, the extra sensing information can be used to decrease the sampling period up to $h = \tau/2 = 0.042\ s$. The system response from both controllers is shown in Fig. 2. The pipelined controller shows a settling time significantly shorter than the sequential implementation. Notice that the system response remains at zero in both controllers till the elapsed time is larger than τ. This is because the image-processing algorithm has not delivered the first output; therefore no sensing information is available to the controller.

3.2 Modelling pipelined systems

This subsection summarizes the modelling strategy presented in [29] for performance-oriented pipelined sensing controllers.

Given the continuous-time system with sensing-to-actuating delay τ:

$$\begin{aligned} \dot{x}(t) &= A_c x(t) + B_c u(t), t \in [kh + \tau, (k+1)h + \tau) \\ y(t) &= C_c x(t) \end{aligned} \tag{2}$$

where $A_c \in \mathbb{R}^{n \times n}$, $B_c \in \mathbb{R}^n$, and $C_c \in \mathbb{R}^{1 \times n}$ are state, input, and output matrices, $x \in \mathbb{R}^n$ and $u \in \mathbb{R}$ are state and input vectors and $k = \{0, 1, 2, \ldots\}$. $u(t)$ is piecewise constant in the intervals $t \in [kh + \tau, (k+1)h + \tau]$. τ is a constant sensor-to-actuator delay. h is the sampling period.

For a resource configuration with γ sensing cores, the sampling period is defined by:

$$h = \frac{\tau}{\gamma}. \tag{3}$$

Such a sampling period produces a discrete-time pipelined system of the form:

$$z((k+1)h) = \Phi_d z(kh) + \Gamma_d u(kh), \tag{4}$$

with $z \in \mathbb{R}^{(n+\gamma)}$ the augmented state vector, $\Phi_d \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$ and $\Gamma_d \in \mathbb{R}^{(n+\gamma)}$ the nominal augmented discrete-time state and input matrices respectively. The augmented state is defined by:

$$z(kh) = \begin{bmatrix} x^T(kh) & u((k-\gamma)h) & u((k-\gamma+1)h) & \cdots & u((k-2)h) & u((k-1)h) \end{bmatrix}^T. \quad (5)$$

The discrete-time input and state matrices are defined by:

$$\Phi_d = \begin{bmatrix} A_d & B_d & 0_{n\times(\gamma-i)} \\ 0_{(\gamma-i)\times n} & 0_{(\gamma-i)\times i} & I_{(\gamma-i)\times(\gamma-i)} \\ 0_{i\times n} & 0_{i\times i} & 0_{i\times(\gamma-i)} \end{bmatrix}, \Gamma_d = \begin{bmatrix} 0_{n\times i} \\ 0_{(\gamma-i)\times i} \\ I_{i\times i} \end{bmatrix} \quad (6)$$

with $A_d \in \mathbb{R}^{n\times n}$ and $B_d \in \mathbb{R}^{n\times i}$ the discretized A_c and B_c respectively with sampling period h. Computing A_d and B_d can be done using Taylor series expansion [31].

The control law is defined by:

$$u(kh) = Kz(kh) + Fr \quad (7)$$

where K is the feedback gain, F is the feed-forward gain, and r is the constant reference. For the closed loop representation, we define:

$$\Phi_{cl} = \Phi_d + \Gamma_d K \quad (8)$$

which is used in the robustness analysis of Section 7.

Example 3.1. Using the example pipelined controller of Section 3.1, a resource configuration with two sensing pipes ($\gamma = 2$) results in a sampling period $h = 0.042\ s$ and an augmented state vector $z(kh) = [z_1(k)\ z_2(k)\ u((k-2)h)\ u((k-1)h)]^T$. z_1 and z_2 are the discrete-time equivalent of x_1 and x_2 respectively. The discrete-time augmented matrices are then given by:

$$\Phi_d = \begin{bmatrix} 0.986 & 0.070 & 0.015 & 0 \\ -0.358 & 0.886 & 0.397 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \Gamma_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

3.3 Control design and trade-off analysis

This subsection summarizes the method described in [30] for tuning a performance-oriented controller with pipelined sensing.

To analyse the trade-off between resource usage and QoC, a controller with optimized performance has to be tuned for each resource configuration. Performance metrics related to the response of the controller (e.g. tracking velocity, settling time) are of interest in many real-life applications, e.g. [22, 23, 29]. We consider the settling time as a performance metric, because it is relevant in most real-time control applications. We then define our QoC performance as:

$$QoC = \frac{1}{S_t}$$

where S_t is the controller settling time. Settling time is considered as the time that the system output takes to reach and stay within a bound of 2% around the reference r, when a reference change occurs. The feedback gain K of Eq. 7 is designed using the well known Linear Quadratic Regulator (LQR). LQR finds a K that minimizes the quadratic cost:

$$J = \min \sum_{kh=0}^{\infty} z^T(kh)Hz(kh) + u^T(kh)Ru(kh) \quad (9)$$

with $H \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$ and $R \in \mathbb{R}$, the state and input weight matrices of the LQR, which are symmetric positive semi-definite and positive definite matrices respectively, i.e. $H = H^T \succeq 0$ and

$R = R^T > 0$ [25]. The feed-forward gain F of Eq. 7 can then be designed according to a set-point regulation equation [29]:

$$F = \begin{bmatrix} K & I \end{bmatrix} \begin{bmatrix} \Phi_d - I & \Gamma_d \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix}$$

Finding F can be straightforwardly done once K is available. However, for finding K, note that the cost function of Eq. 9 gives a controller which is optimal with respect to J in terms of the tuning parameters H and R; this does not necessarily mean that it has the shortest settling time. Finding H and R that minimize the settling time is referred to as LQR tuning.

In [30] and this work, PSO is used to tune an LQR for minimum settling time. A PSO algorithm employs a swarm of m particles X_q, with $q = 1 \ldots m$ to explore the design space of a problem in order to minimize a fitness metric. In this case, the design space corresponds to the values of H and R and the fitness metric corresponds to the controller settling time. The challenge arises because H and R have to remain positive (semi-) definite while they are explored by the PSO algorithm. To address this, the PSO algorithm presented in [30] uses two intermediate matrix variables Ψ_q and Ω_q to define the swarm i.e. $X_q = [\Psi_q \ \Omega_q]$. The tuning parameters satisfy:

$$H_q = \Psi_q^T \Psi_q, R_q = \Omega_q^T \Omega_q \tag{10}$$

Eq. 10 guarantees that H_q and R_q are positive (semi-) definite. PSO can then be used to find the values of Ψ_q and Ω_q that minimize the settling time for a resource configuration. This tuning technique can be applied to multiple resource configurations in order to find the number of pipes that still gives a meaningful improvement on QoC i.e., an improvement that has an impact on application performance.

Example 3.2. Following Example 3.1, the PSO particle swarm population is defined by:

$$X_q = \begin{bmatrix} \Psi_{11q} & \Psi_{12q} & \Psi_{13q} & \Psi_{14q} & \ldots & \Psi_{44q} & \Omega_{1q} \end{bmatrix}$$

where

$$\Psi_q = \begin{bmatrix} \Psi_{11q} & \ldots & \Psi_{14q} \\ \vdots & \ddots & \vdots \\ \Psi_{41q} & \ldots & \Psi_{44q} \end{bmatrix}, \Omega_q = \Omega_{1q}$$

The PSO algorithm can then explore Ψ_q and Ω_q that minimize the settling time. In this example, we use a swarm with $m = 100$ particles, resulting in a settling time $S_t = 123.5\ ms$ and controller:

$$K = \begin{bmatrix} -30.9 & -7.3 & -2.7 & -1.9 \end{bmatrix}, F = 36.1.$$

PSO is applied to multiple resource configurations to analyse the trade-off between processing resources and settling time shown in Fig. 3. This figure shows that each newly added pipe reduces the settling time. We consider a reduction of the settling time of at least 2 ms meaningful. The improvement from five to eight pipes is less , which suggests four pipes as the resource configuration.

The method discussed in this subsection is used to explore the trade-off between the number of pipes and settling time. However, in most applications it is common to have modelling uncertainties, which combined with the pipelined delay potentially affect the system stability and the trade-off analysis. It is then necessary to analyse the robustness of the designed controller.

4 OVERVIEW OF THE PROPOSED DESIGN FLOW

To analyse the robustness of a performance-oriented pipelined controller or to design a robustness-oriented controller, the following design flow is proposed, clarified in Fig. 4.

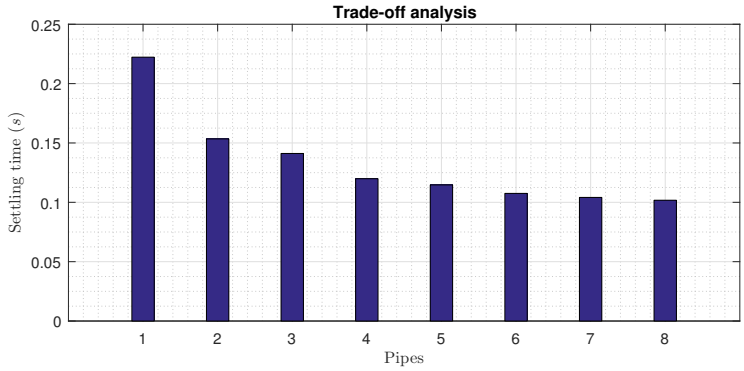

Fig. 3. Trade-off of the example of Section 3.1, without considering model uncertainties.

(1) **Discretize model**: given a continuous-time nominal dynamic model, a sensing delay, and a set of resource configurations (i.e. the number of sensing cores used), a discrete-time nominal pipelined model is found for each resource configuration (as outlined in Section 3.2).
(2) **Performance-oriented controller design**: given the discrete-time nominal pipelined models for each resource configuration. We propose to use PSO to tune a controller with minimum settling time as an optimization objective (as outlined in Section 3.3). Note that the controller is designed using the nominal plant. We subsequently analyse if the designed controller is robust against given uncertainties in step four.
(3) **Benchmark uncertain model**: given continuous-time additive uncertainties for the dynamic model, the set of resource configurations, and the sensing delay, discrete-time additive uncertainties are found for each resource configuration. Section 5 presents a discrete-time pipelined uncertain model while Section 6 describes the benchmarking of the discrete-time uncertainties. The benchmarking for the basis for analysis (Steps (4) and (5)) or design (Step (6))If a robustness-oriented pipelined controller is desired, step six is implemented. If the

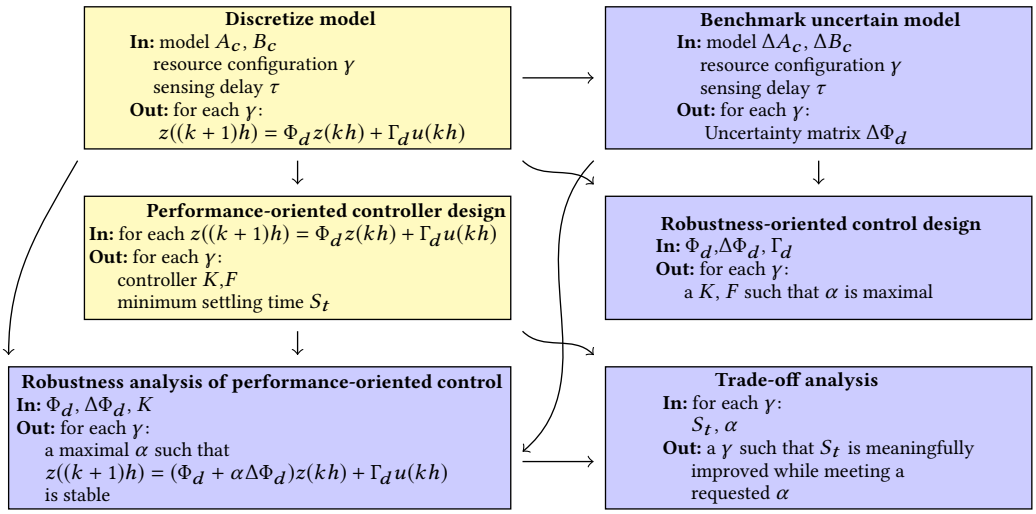

Fig. 4. Overview of the proposed design flow.

robustness analysis of a performance-oriented control is desired, steps four and five are applied.

(4) **Analyse robustness of performance-oriented control**: given the discrete-time controller, nominal and uncertain model, we analyse the robustness of each resource configuration. To do so, we parametrize the uncertainties as a scalar α, which is maximized for guaranteed stability. Section 7.1 presents basic control robustness theory while the details of the robustness analysis are presented in Section 7.2.

(5) **Trade-off analysis**: given the settling times and the robustness analysis (i.e. an α) for each resource configuration, a resource configuration is chosen such that the settling time is still meaningfully improved while a robustness constraint (i.e. a minimum desired α) is met. Section 8 shows a detailed example of this analysis.

(6) **Robustness-oriented control design**: given the discrete-time uncertain pipelined model, a controller is designed with maximal robustness for each resource configuration. To do so, we find controllers that maximize the parametrized scalar α. Details of are given in Section 7.3.

5 MODELLING PIPELINED SYSTEMS WITH UNCERTAINTIES

To perform a robust analysis of the discrete-time pipelined controller, the model with uncertainties has to be first discretized. To do so, this section describes the discretization of continuous-time bounded additive uncertainties. The resulting discrete-time uncertainties are augmented according to the structure of pipelined systems.

5.1 Continuous-time model with additive uncertainties

Consider that the continuous-time system described in Eq. 2 contains additive uncertainties:

$$\dot{x}(t) = \bar{A}_c(t)x(t) + \bar{B}_c(t)u(t), t \in [kh + \tau, (k+1)h + \tau) \tag{11}$$

where:

$$\bar{A}_c(t) = A_c + \Delta A_c(t), \bar{B}_c(t) = B_c + \Delta B_c(t), \tag{12}$$

with A_c, B_c, C_c, $x(t)$, $u(t)$ were introduced in the model of Eq. 2. $\Delta A_c(t) \in \mathbb{R}^{n\times n}$, $\Delta B_c(t) \in \mathbb{R}^n$ are continuous-time time-varying model uncertainties.

5.2 Discrete-time equivalent of continuous-time uncertainties

Discretizing Eq. 11 with sampling period h and sensor-to-actuator delay τ using Eq. 3, we obtain,

$$x((k+1)h) = (A_d + \Delta A_d(kh))x(kh) + (B_d + \Delta B_d(kh))u((k-\gamma)h), \tag{13}$$

where $A_d \in \mathbb{R}^{n\times n}$, $B_d \in \mathbb{R}^n$ are as per Eq. 6 while $\Delta A_d(kh) \in \mathbb{R}^{n\times n}$ and $\Delta B_d(kh) \in \mathbb{R}^n$ are the resulting discretized time-varying model uncertainties. The benchmarking of the time-varying matrices is later explained in Section 6.

5.3 Augmented model for the pipelined system with uncertainties

First, we define the augmented states using Eq. 5. Considering the discretized model in Eq. 13, we obtain the following augmented system:

$$z((k+1)h) = \bar{\Phi}_d(kh)z(kh) + \Gamma_d u(kh) \tag{14}$$

$$y(kh) = C_d z(kh), \tag{15}$$

with

$$\bar{\Phi}_d(kh) = \Phi_d + \Delta\Phi_d(kh),$$

where Φ_d and Γ_d were defined in the model of Eq. 6. $\Delta\Phi_d(kh)$ is the discrete-time augmented uncertainty matrix given by:

$$\Delta\Phi_d(kh) = \begin{bmatrix} \Delta A_d(kh) & \Delta B_d(kh) & 0_{n\times(\gamma-1)} \\ 0_{\gamma\times n} & 0_{\gamma\times 1} & 0_{\gamma\times(\gamma-1)} \end{bmatrix}, \tag{16}$$

where I and 0 denote identity and zero matrices of appropriate dimensions. Note that Γ_d is not influenced by the model uncertainties.

5.4 Structure of the model uncertainties

Model uncertainties considered in this work are norm-bounded and of the following form:

$$\Delta\Phi_d(kh) = \alpha D G(kh) E \tag{17}$$

$D \in \mathbb{R}^{(n+\gamma)\times n}$, $E \in \mathbb{R}^{n\times(n+\gamma)}$ are known constant matrices while $G(kh) \in \mathbb{R}^{n\times n}$ is an uncertain time-varying matrix:

$$G^T(kh)G(kh) \leq I. \tag{18}$$

$\alpha \in \mathbb{R}^+$ is a positive constant which is used as a scaling factor later in the design. A greater α implies higher robustness.

The method presented in this section provides a strategy to model discrete-time norm bounded additive uncertainties in pipelined systems. The resulting discrete-time uncertainties are then used to analyse the system robustness in Section 7.

6 BENCHMARKING THE MODEL UNCERTAINTY

The structure of the model uncertainties described in Section 5.4 requires knowledge of the constant matrices $D \in \mathbb{R}^{(n+\gamma)\times n}$, $E \in \mathbb{R}^{n\times(n+\gamma)}$ and the time-varying matrix $G(kh) \in \mathbb{R}^{n\times n}$ as shown in Eq. 18. It can be noted from Theorem 7.5 (Robustness analysis) and Theorem 7.7 (Robust design) presented later that the choice for the D and E matrices influences the analysis outcome. Intuitively, these two matrices define the boundaries of the uncertainty matrix $\Delta\Phi_d(kh)$. In the following subsections, we benchmark these matrices based on the boundaries of the uncertainty matrix.

6.1 Uncertainty bound

The time-varying uncertainty $\Delta\Phi_d(kh)$ described in Section 5.4 has a constraint given by $G^T(kh)G(kh) \leq I$. Notice that this constraint also implies that $||G(kh)|| \leq 1$. We derive the upper bound on the uncertainty as:

$$\begin{gathered} ||\Delta\Phi_d(kh)|| = ||\alpha D G(kh) E|| \\ ||\Delta\Phi_d(kh)|| \leq ||\alpha||\ ||D||\ ||G(kh)||\ ||E|| \leq ||\alpha||\ ||D||\ ||E|| \\ ||\Delta\Phi_d(kh)|| \leq ||\alpha||\ ||D||\ ||E||. \end{gathered} \tag{19}$$

Eq. 19 provides an upper bound on the maximum norm of uncertainties that can be tolerated by the analysis and design method presented in the next section. This upper bound depends on α, D, and E. α is maximized during the optimization analysis to obtain the maximal upper bound on the uncertainties. D and E are constant matrices that need to be computed before the optimization procedure. Note that an incorrect selection of these matrices (e.g. matrices with smaller norm than the real one) can be compensated by α during the optimization analysis to obtain the maximal upper bound. For example, for a D and E with a smaller norm, a larger α should be found. In the following subsections, we propose an estimation technique to select D and E.

6.2 Selection of D and E

Let us define the terms $\Delta A_{d,wc}$ and $\Delta B_{d,wc}$ in Eq. 16, as the $\Delta A_d(kh)$ and $\Delta B_d(kh)$ with the maximum norm over the range of uncertainty under consideration:

$$\begin{aligned} ||\Delta A_{d,wc}|| &= \max_k ||\Delta A_d(kh)||, \\ ||\Delta B_{d,wc}|| &= \max_k ||\Delta B_d(kh)||. \end{aligned} \tag{20}$$

Obviously, $\Delta A_{d,wc}$ and $\Delta B_{d,wc}$ are dependent on $\Delta A_c(t)$ and $\Delta B_c(t)$. We propose the following structure of the D and E matrices,

$$D = \begin{bmatrix} I_{n\times n} \\ 0_{\gamma\times n} \end{bmatrix}, E = \begin{bmatrix} \Delta A_{d,wc} & \Delta B_{d,wc} & 0_{n\times(\gamma-1)} \end{bmatrix}. \tag{21}$$

Including the definition of D and E of Eq. 17, we obtain:

$$\Delta\Phi_d(kh) = \alpha DG(kh)E = \begin{bmatrix} \alpha G(kh)\Delta A_{d,wc} & \alpha G(kh)\Delta B_{d,wc} & 0_{n\times(\gamma-1)} \\ 0_{\gamma\times n} & 0_{\gamma\times 1} & 0_{\gamma\times(\gamma-1)} \end{bmatrix}, \tag{22}$$

Eq. 22 defines the terms $\Delta A_d(kh)$ and $\Delta B_d(kh)$ in Eq. 16. Eq. 22 further depends on $\Delta A_{d,wc}$ and $\Delta B_{d,wc}$ which are benchmarked in the following.

6.3 Computation of $\Delta A_{d,wc}$ and $\Delta B_{d,wc}$

In practice, computation of exact $\Delta A_{d,wc}$ and $\Delta B_{d,wc}$ is hard since it requires evaluation of all (infinitely many) possible combinations of $\Delta A_c(t)$ and $\Delta B_c(t)$. We therefore rely on an experimental method for a restricted class of model uncertainty for benchmarking. We consider the case where only one element of $\Delta A_c(t)$ and $\Delta B_c(t)$ is non-zero with given minimum and maximum values. That is, the uncertain element $a(t)$ of $\Delta A_c(t)$ is bounded by $a_{min} \leq a(t) \leq a_{max}$. Similarly, the uncertain element $b(t)$ of $\Delta B_c(t)$ is bounded by $b_{min} \leq b(t) \leq b_{max}$. For this case, we benchmark $||\Delta A_{d,wc}||$, $||\Delta B_{d,wc}||$ as:

$$\begin{aligned} ||\Delta A_{d,wc}|| = \quad & \max ||\Delta A_d||, \\ & a \in \{a_{min}, a_{max}\} \\ ||\Delta B_{d,wc}|| = \quad & \max ||\Delta B_d|| \\ & a \in \{a_{min}, a_{max}\} \\ & b \in \{b_{min}, b_{max}\} \end{aligned} \tag{23}$$

where ΔA_d and ΔB_d are constant discretized uncertainties which are computed with the following two propositions. These propositions require constant ΔA_c and ΔB_c i.e. constant values of a, b. Therefore, we assign constant values to a and b in the range $a_{min} \leq a(t) \leq a_{max}$, $b_{min} \leq b(t) \leq b_{max}$ to obtain a set of ΔA_d, ΔB_d. We then select $\Delta A_{d,wc}$ and $\Delta B_{d,wc}$ using Eq. 23.

Proposition 6.1. ***Computation of*** ΔA_d ***for a given*** ΔA_c***:*** *Consider the uncertain systems in Eq. 11 and Eq. 13. A constant* ΔA_d *for a given* ΔA_c *is computed by:*

$$\Delta A_d = e^{(A_c+\Delta A_c)h} - e^{A_c h} \tag{24}$$

Proposition 6.2. ***Computation of*** ΔB_d ***for given*** ΔA_c ***and*** ΔB_c*: Consider the uncertain systems in Eq. 11 and Eq. 13.* ΔB_d *for constant* ΔA_c *and* ΔB_c *is computed by:*

$$B_d = \int_0^h e^{(A_c+\Delta A_c)s}(B_c + \Delta B_c)ds - \int_0^h e^{\Delta A_c s}(\Delta B_c)ds \tag{25}$$

Example 6.3. Using the example of Section 3.1, we define the uncertainties as:

$$\Delta A_c(t) = \begin{bmatrix} 0 & 0 \\ 0 & a(t) \end{bmatrix}, \Delta B_c(t) = \begin{bmatrix} 0 \\ b(t) \end{bmatrix}$$

where

$$-0.05 \leq a(t) \leq 0.05$$
$$-0.2 \leq b(t) \leq 0.2$$

$a(t)$ and $b(t)$ correspond to an uncertainty of 2% in one parameter in the nominal matrices. Propositions 6.1 and 6.2 are used to find one ΔA_d and one ΔB_d for a range of constant continuous-time uncertainties $\Delta A_c(t)$ and $\Delta B_c(t)$. The continuous-time uncertainties that produce the discrete-time matrices with the largest norm are:

$$\Delta A_c = \begin{bmatrix} 0 & 0 \\ 0 & 50 \end{bmatrix} \times 10^{-3}, \Delta B_c = \begin{bmatrix} 0 \\ 200 \end{bmatrix} \times 10^{-3} \tag{26}$$

The uncertainties with the worst-case norm are then:

$$\Delta A_{d,wc} = \begin{bmatrix} -9.3 & 72.7 \\ -370.8 & 1879 \end{bmatrix} \times 10^{-6}, \Delta B_{c,wc} = \begin{bmatrix} 312.3 \\ 8379.3 \end{bmatrix} \times 10^{-6}$$

For the case of two pipes ($\gamma = 2$), then we use Eq. 21:

$$E = \begin{bmatrix} -9.3 & 72.7 & 312.3 & 0 \\ -370.8 & 1879 & 8379.3 & 0 \end{bmatrix} \times 10^{-6}, D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

7 ROBUSTNESS OF PIPELINED SYSTEMS

We use a control law of the form shown in Eq. 7. The gains K and F need to be analysed or redesigned for the model with uncertainties described in Section 5.4 for guaranteeing closed-loop stability. Note that F does not influence stability but it is important for performance benchmarking experiments which are performed for a given robust feedback gain K. In the following, we focus on analyzing and/or designing feedback gain K with respect to the model uncertainties described above to guarantee closed-loop stability.

Using $u(kh) = Kz(kh)$, the closed-loop model for the system Eq. 14:

$$z((k+1)h) = \bar{\Phi}_{cl}(kh)z(kh) \tag{27}$$

with $\bar{\Phi}_{cl} \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$ given by

$$\bar{\Phi}_{cl}(kh) = \Phi_{cl} + \Delta\Phi_d(kh), \tag{28}$$

where Φ_{cl} is the nominal closed-loop system further given by

$$\Phi_{cl} = \Phi_d + \Gamma_d K. \tag{29}$$

In the following, we focus on the following questions:

- **Robustness analysis for a performance-oriented controller:** given a feedback gain K that stabilizes the nominal closed-loop system in Eq. 29, find the maximum α for which closed-loop stability of the uncertain system in Eq. 27 can be guaranteed.
- **Robustness-oriented controller design:** design a feedback gain K maximizing α such that closed-loop stability of the uncertain system in Eq. 27 can be guaranteed.

To analyse robustness or redesign a pipelined control system against additive norm-bounded time-varying uncertainties, Lemmas 7.1 to 7.4 bellow are needed. Then, Theorem 7.5 presents a robustness analysis technique for an already designed controller and Theorem 7.7 presents a robust control design technique. These theorems are adapted from [13, 40] to pipelined systems.

7.1 Preliminary lemmas

LEMMA 7.1. *[43] Given the constant matrices Θ, Λ and Ξ of appropriate dimensions with $\Theta = \Theta^T$, and the time-varying matrix $G(kh)$, the following inequality holds:*

$$\Theta + \Lambda G(kh)\Xi + (\Lambda G(kh)\Xi)^T \prec 0$$

for all $G(kh)$ satisfying $G^T(kh)G(kh) \prec I$, if and only if there exists a scalar $\epsilon \in \mathbb{R}^+$ such that

$$\Theta + \epsilon^{-1}\Lambda\Lambda^T + \epsilon\Xi^T\Xi \prec 0.$$

LEMMA 7.2. ***Schur complement [46]:*** *Given the matrices M, L, and P, with $M = M^T$, and the positive definite matrix $P = P^T \succ 0$, the matrix inequality $M + L^T P^{-1} L \prec 0$ can be rewritten in the following form:*

$$\begin{bmatrix} M & L^T \\ L & -P \end{bmatrix} \prec 0. \tag{30}$$

LEMMA 7.3. ***Discrete-time stability [32, Chapter 23]:*** *The system in Eq. 27 is stable if there exists a positive definite matrix $P \in \mathbb{R}^{(n+\gamma_{PC})\times(n+\gamma_{PC})}$ such that the following inequality holds for all $\bar{\Phi}_{cl}(kh)$:*

$$\bar{\Phi}_{cl}^T(kh) P \bar{\Phi}_{cl}(kh) - P \prec 0.$$

Finally, we introduce Lemma 7.4 since it is shown to be less conservative than Lemma 7.3 in [8, 12].

LEMMA 7.4. *[12] The system in Eq. 27 is stable if there exist a matrix $O \in \mathbb{R}^{(n+\gamma_{PC})\times(n+\gamma_{PC})}$ such that $O = O^T \succ 0$ (i.e., positive definite), and a matrix $V \in \mathbb{R}^{(n+\gamma_{PC})\times(n+\gamma_{PC})}$ such that*

$$\begin{bmatrix} O & \bar{\Phi}_{cl}(kh)V \\ (\bar{\Phi}_{cl}(kh)V)^T & V^T + V - O \end{bmatrix} \succ 0, \tag{31}$$

for all $\bar{\Phi}_{cl}(kh)$.

7.2 Robustness analysis for performance oriented controller

The following theorem finds the maximum scalar α that stabilizes the system of Eq. 27 with a given feedback gain K. α is therefore used to quantify the controller robustness for a particular resource configuration. A larger α implies a higher system robustness.

THEOREM 7.5. ***Robustness analysis:*** *Consider the closed-loop uncertain system (27) where $\bar{\Phi}_{cl}(kh) \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$, the nominal closed-loop system (29) is stable and $\Delta\Phi_d(kh) \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$ is given by Eq. 17 and Eq. 18 with known matrices $D \in \mathbb{R}^{(n+\gamma)\times n}$, $E \in \mathbb{R}^{n\times(n+\gamma)}$. The system in Eq. 27 is stable if there exist a positive definite matrix $O \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$, a matrix $V \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$, and a scalar $\lambda \in \mathbb{R}^+$ that*

$$\textbf{\textit{minimize}}\ \sigma$$

$$\textbf{\textit{subject to}} \begin{bmatrix} -O & \lambda D & -\Phi_{cl}V & 0 \\ \lambda D^T & -\sigma\lambda I & 0 & 0 \\ -(\Phi_{cl}V)^T & 0 & O - V - V^T & (EV)^T \\ 0 & 0 & (EV) & -\lambda I \end{bmatrix} \prec 0$$

with $\sigma = \alpha^{-2}$.

Proof. For stability of system (27), we start with using Lemma 7.4:

$$\begin{bmatrix} O & \bar{\Phi}_{cl}(kh)V \\ (\bar{\Phi}_{cl}(kh)V)^T & V^T + V - O \end{bmatrix} \succ 0$$

Substituting $\bar{\Phi}_{cl}(kh)$ in Eq. 28, using Eq. 17 and multiplying by -1 yields:

$$\begin{bmatrix} -O & -\Phi_{cl}V - \alpha DG(kh)EV \\ -(\Phi_{cl}V)^T - \alpha\,(DG(kh)EV)^T & O - V - V^T \end{bmatrix} \prec 0$$

$$\Rightarrow \begin{bmatrix} -O & -\Phi_{cl}V \\ -(\Phi_{cl}V)^T & O - V - V^T \end{bmatrix} + \begin{bmatrix} 0 & -\alpha DG(kh)EV \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\alpha\,(DG(kh)EV)^T & 0 \end{bmatrix} \prec 0$$

Double transposing the third matrix above yields:

$$\begin{bmatrix} -O & -\Phi_{cl}V \\ -(\Phi_{cl}V)^T & O - V - V^T \end{bmatrix} + \begin{bmatrix} 0 & -\alpha DG(kh)EV \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -\alpha DG(kh)EV \\ 0 & 0 \end{bmatrix}^T \prec 0$$

Decomposing the second and third matrices gives:

$$\begin{bmatrix} -O & -\Phi_{cl}V \\ -(\Phi_{cl}V)^T & O - V - V^T \end{bmatrix} + \begin{bmatrix} -\alpha D \\ 0 \end{bmatrix} G(kh) \begin{bmatrix} 0 & EV \end{bmatrix} + \left\{ \begin{bmatrix} -\alpha D \\ 0 \end{bmatrix} G(kh) \begin{bmatrix} 0 & EV \end{bmatrix} \right\}^T \prec 0$$

Applying Lemma 7.1, we obtain:

$$\begin{bmatrix} -O & -\Phi_{cl}V \\ -(\Phi_{cl}V)^T & O - V - V^T \end{bmatrix} + \epsilon^{-1} \begin{bmatrix} -\alpha D \\ 0 \end{bmatrix} \begin{bmatrix} -\alpha D^T & 0 \end{bmatrix} + \epsilon \begin{bmatrix} 0 \\ (EV)^T \end{bmatrix} \begin{bmatrix} 0 & EV \end{bmatrix} \prec 0$$

Note that in the above condition, the second and third element capture the influence of the uncertainty on the overall system stability. These terms depend on the free matrices D, E, and V. Therefore it is not possible to generalize their negative or positive definiteness. The first term captures the influence on the stability of the nominal closed-loop system. Since the closed-loop nominal system is stable, this term must be negative definitive as shown in Lemma 7.4. Further,

$$\begin{bmatrix} -O & -\Phi_{cl}V \\ -(\Phi_{cl}V)^T & O - V - V^T \end{bmatrix} + \begin{bmatrix} \epsilon^{-1}\alpha^2 DD^T & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \epsilon(EV)^T EV \end{bmatrix} \prec 0$$

$$\Rightarrow \begin{bmatrix} -O + \epsilon^{-1}\alpha^2 DD^T & -\Phi_{cl}V \\ -(\Phi_{cl}V)^T & O - V - V^T + \epsilon(EV)^T EV \end{bmatrix} \prec 0 \tag{32}$$

Applying the Schur complement (Lemma 7.2) with respect to the term $-O + \epsilon^{-1}\alpha^2 DD^T$:

$$\begin{bmatrix} -O & \epsilon^{-1}D & -\Phi_{cl}V \\ \epsilon^{-1}D^T & -\epsilon^{-1}\alpha^{-2}I & 0 \\ -(\Phi_{cl}V)^T & 0 & O - V - V^T + \epsilon(EV)^T EV \end{bmatrix} \prec 0$$

Applying the Schur complement on the term $O - V - V^T + \epsilon(EV)^T EV$:

$$\begin{bmatrix} -O & \epsilon^{-1}D & -\Phi_{cl}V & 0 \\ \epsilon^{-1}D^T & -\epsilon^{-1}\alpha^{-2}I & 0 & 0 \\ -(\Phi_{cl}V)^T & 0 & O - V - V^T & (EV)^T \\ 0 & 0 & EV & -\epsilon^{-1}I \end{bmatrix} \prec 0$$

Defining $\lambda = \epsilon^{-1}$ and $\sigma = \alpha^{-2}$, the above is equivalent to Theorem 7.5 completing the proof. By finding the minimum σ satisfying the above LMI, we obtain the maximum α (scaling factor of robustness). □

Note that in Theorems 7.5 and 7.7, minimizing σ maximizes α, which means that the uncertainties that can be handled are also maximized.

Remark 1. *The conditions shown in Theorem 7.5 (and also Theorem 7.7) correspond to a non-linear matrix inequality due to the multiplication of the terms σ and λ. However, assuming a value of $\lambda > 0$, the conditions are simplified to a Linear Matrix Inequality. Therefore, we assign different values to λ between a small number (e.g. 100×10^{-27}) and a large number (e.g. 10) that produce a feasible solution. We solve each of the resulting optimization problems and we then report the largest α. This strategy does not guarantee that the found α is optimal due to the discretization of the design space of λ.*

Remark 2. *Applying the Schur complement (Lemma 7.2) to one element of a block matrix (e.g. Eq. 32) divides one element into four new elements. As a result, a new row and a new column are created inside the matrix inequality which are filled by the newly created elements and zeros. The organization of such elements might differ as long as they are placed according to the symmetry of the matrix inequality. It is possible to prove that two matrix inequalities L_1 and L_2 with elements placed symmetrically in different positions are equivalent by finding a transformation matrix F such that $F^T L_1 F = L_2$.*

Example 7.6. Consider the model from Example 3.1, the controller from Example 3.2, and the following uncertainties:

$$E = \begin{bmatrix} -9.3 & 72.7 & 312.3 & 0 \\ -370.8 & 1879 & 8379.3 & 0 \end{bmatrix} \times 10^{-6}, D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

The E and D matrices are benchmarked with the procedure explained in Section 9. Consider also that the minimum robustness required for this example is $\alpha \geq 1$. This implies that the controller can tolerate uncertainties at least larger than the product of E and D. The optimization problem of Theorem 7.5 is then formulated.

Using Remark 1, the optimization problem is converted into a set of Linear Matrix Inequalities. The modelling tool Yalmip [26] together with the convex optimization software SDPT3 [35] are used to solve the optimization problem. As a result, $\alpha = 7.7$ is found for $\gamma = 2$. Given that $\alpha \geq 1$, the robustness constraint of our problem is met for two pipes.

7.3 Robustness-oriented pipelined controller design

The following theorem finds the feedback gain K that maximizes the scalar α, while guaranteeing the stability of Eq. 27. Like in the previous case, a larger α implies a more robust controller.

Theorem 7.7. ***Robust controller design:*** *Consider the closed-loop uncertain system (27) where $\bar{\Phi}_{cl}(kh) \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$ and $\Delta\Phi_d(kh)$ are given by Eq. 17 and Eq. 18 with known matrices $D \in \mathbb{R}^{(n+\gamma)\times n}$, $E \in \mathbb{R}^{n\times(n+\gamma)}$. A robust feedback gain $K \in \mathbb{R}^{1\times(n+\gamma)}$ exists if there exist a positive definite matrix $O \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$, matrices $V \in \mathbb{R}^{(n+\gamma)\times(n+\gamma)}$ and $V_k \in \mathbb{R}^{1\times(n+\gamma)}$, and scalar $\lambda \in \mathbb{R}^+$ that*

$$\textbf{\textit{minimize}}\ \sigma$$

$$\textbf{\textit{subject to}} \begin{bmatrix} -O & \lambda D & -\Phi_d V - \Gamma_d V_k & 0 \\ \lambda D^T & -\sigma\lambda I & 0 & 0 \\ -(\Phi_d V + \Gamma_d V_k)^T & 0 & O - V - V^T & (EV)^T \\ 0 & 0 & (EV) & -\lambda I \end{bmatrix} \prec 0$$

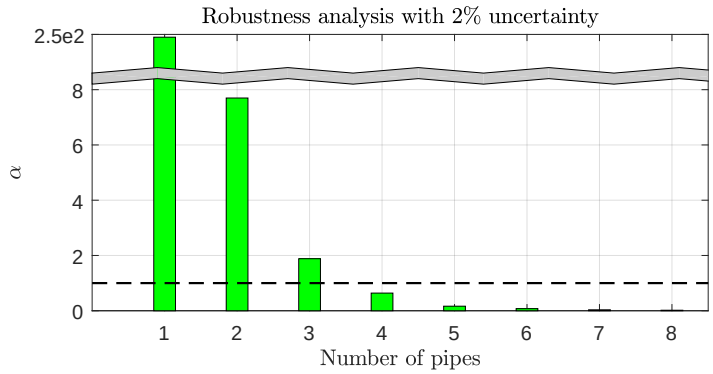

Fig. 5. Robustness analysis of motivational example.

with $V_k = KV$ *and* $\sigma \in \mathbb{R}^+$ *and* $K = V_k V^{-1}$.

Proof. The proof follows the steps of Theorem 7.5 using the definition of Φ_{cl} (Eq. 29) and $V_k = KV$. The details are therefore omitted. The invertibility of V is implied because from the matrix inequality it can be deduced that $V + V^T \succ O \succ 0$, which means that V is full rank and therefore invertible. □

8 ROBUSTNESS EXAMPLE OF PIPELINED SYSTEMS

The robustness analysis procedure of Section 7 shows to quantify the robustness of a pipelined controller by means of the scalar α. However, in pipelined control, each resource configuration has a different discrete-time uncertainties model. Therefore, the robustness analysis of a performance-oriented controller or the design of a robustness-oriented controller has to be repeated for each resource configuration to be considered. This procedure allows for the performance-oriented design to select a resource configuration that improves QoC while meeting a robustness requirement (i.e. minimum α). For the robustness-oriented control design it allows to select a resource configuration that provides a controller with maximized robustness. This is best explained by means of the following examples:

Example 8.1. **Robustness analysis of performance-oriented controller:** continuing with Example 7.6, the robustness analysis is applied to a range of resource configurations from one to eight pipes i.e. $\gamma = 1, \ldots, 8$. Fig. 5 shows the maximum α found in each one of the resource configurations. Taking into account that the robustness requirement is $\alpha \geq 1$, we observe the following from Fig. 5:

- From Example 3.2 it was concluded that the resource configuration that gives a meaningful improvement on QoC is four pipes. However, the robustness constraint is only met in the resource configurations $\gamma = 1, \ldots, 3$ with $\alpha \geq 1$. Therefore, the resource configuration that gives a meaningful improvement on settling time while guaranteeing the robustness constraint is three pipes.
- Each newly added pipe decreases the maximum α that the controller can tolerate, because increasing the number of pipes produces a more aggressive controller response, which is more likely to become unstable with model uncertainties.
- Only a 2% uncertainty in one of the elements of the model matrices has a major impact on the overall system robustness. This is because the B_c matrix has only one non-zero element.

Table 1. Settling times (ms) of motivational example with different values of uncertainties

Number of pipes	$\Delta A_{c(2,2)} = 0$ $\Delta B_{c(2,1)} = 0$	$\Delta A_{c(2,2)} = -0.0025$, $\Delta B_{c(2,1)} = 0.010$	$\Delta A_{c(2,2)} = 0.0025$ $\Delta B_{c(2,1)} = -0.010$
1	222.2	247.5	313.5
2	153.2	165.8	198.7
3	130.5	138.0	161.8
4	119.3	robustness constraint not met, i.e. $\alpha < 1$	
5	111.9		
6	107.5		
7	104.1		
8	101.8		

An uncertainty in such an element directly affects the amount of energy that the controller inputs to the dynamic system, affecting the stability of the system.

- The controller design of Section 3.3 optimizes QoC for a nominal system. However, the uncertain part of the system may deteriorate the QoC. Since the exact value of the uncertainties is not known but only a range, the settling time varies for different values of uncertainties. Table 1 shows an example of such performance deterioration. We used constant continuous-time uncertainties to benchmark these values. The settling time increases with the model uncertainties. However, the best performance is still achieved with the highest number of pipes. The settling times of the systems with uncertainties and resource configurations $\gamma = 4 \ldots 8$ are not shown because $\alpha < 1$ (meaning the robustness constraint is not met).

Example 8.2. **Robustness-oriented pipelined control design** A controller that maximizes α is designed using Theorem 7.7. Fig. 6 shows the resulting α when the initial uncertainty shown in Example 6.3 was increased to 50%. We observe the following from the graphs:

- The maximal α that the system can tolerate increases with the number of pipes, which potentially means that the robustness of the pipelined controller increases with each newly added pipe when designed for maximal robustness.
- The new values of α are significantly larger than the values found with Theorem 7.5, which means a more robust controller is obtained. However, the increased robustness comes at the cost of performance deterioration. For example, in the nominal system with one sensing pipe, the settling time of a controller designed for robustness (using Theorem 7.7) is $2s$, while using the PSO algorithm and not considering the uncertainties in the controller design gives $222ms$.
- The controller resulting from Theorem 7.7 is mostly interesting for systems with large uncertainties, where performance-oriented designs may not be feasible. For example, in ADAS uncertainties originated from changing traffic conditions. There, a more robust controller is desired since it increases vehicle safety. The most robust controller corresponds to the hardware configuration with the largest number of pipes.

9 FEASIBILITY STUDY

9.1 Overview

In this section, we demonstrate the effectiveness of a performance-oriented pipelined control using Hardware-In-the-Loop (HIL) simulation. This simulation consists of emulating the plant behaviour implementing the controller and the model in independent embedded hardware, such that there is an exchange of electrical signals between the plant and the controller (see [3, 16],[27, Chapter 14] and the references therein).

HIL offers the possibility of studying the behaviour of the pipelined controller in a more realistic scenario, e.g. when multiple processing resources exchange their previous controller output to

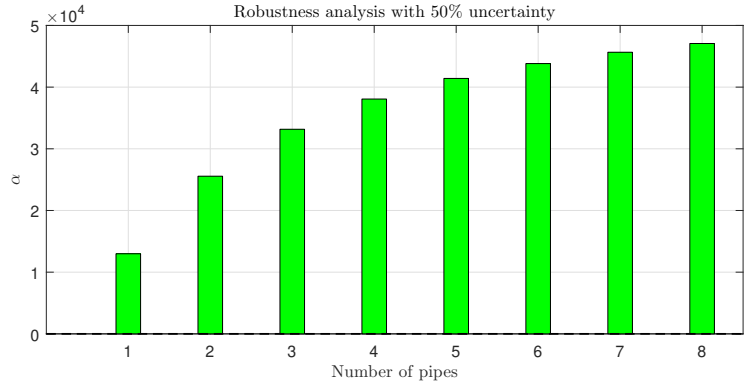

Fig. 6. Robustness analysis including the control design.

compute new controller outputs, or changing the nominal model of the system to evaluate the impact of model uncertainties in the controller.

We subdivide this section in five parts. We introduce an application where pipelined systems can be of benefit. We then present a reconfigurable platform suitable for HIL implementation. Next we describe our HIL simulation. We then apply the design flow of Section 4 considering the available hardware and the presented example. We finally present some results of our HIL simulation.

9.2 Plant description and model

We consider an example of an assembly line like the xCPS system shown in Fig. 7a [1]. This kind of system mimics the behaviour of complex machines such as assembly lines or wafer scanners, in which data-intensive algorithms (e.g. image-processing) are used as sensors in control loops.

In xCPS, the speed of assembly objects travelling on the belt has to be regulated such that the objects move slow when they have to be processed by one of the actuators in front of the belts, and fast when they have to travel between actuators. The conveyor belt rotation speed ω is controlled by a continuous-time PI controller. The settling time of such a controller is 20 ms. The PI controller does not receive information about the object speed moving on top of the conveyor belt, but only about the belt itself. Ideally, both speeds (i.e. belt and object) should be the same, however in practise due to disturbances (e.g. friction of the block with the belt borders) they might differ. Therefore, a camera and an image-processing algorithm are used as a sensor of an additional image-based controller to regulate the block's speed. The Hough transform for circles is used to recognize the position of the block on the belt, which is used to estimate the block speed. A schematic of the system is shown in Fig. 7b. The plant to be controlled by the image-based control is the PI controller together with the model of the conveyor belt (i.e. the inner loop). The output of the image-based controller is then used as reference for the PI controller.

Using the worst-case execution time of the image-processing algorithm, the control computation tasks, and the actuation task, a $\tau = 90ms$ latency is introduced to the image-based control loop, which compared to the $20ms$ settling time of the inner loop, potentially limits the performance of the image-based control loop. A pipelined performance-oriented design may improve the performance of the image-based controller and the system performance.

The conveyor belt is moved by a DC motor. The combined model of such a motor and the PI controller (i.e. the inner loop) are given by:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -K_m^2 - \frac{b_m}{J_m} - \frac{K_p K_m}{R_m J_m} & \frac{K_m K_i}{R_m J_m} \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{K_p K_m}{R_m J_m} \\ 1 \end{bmatrix} r_{PI}$$

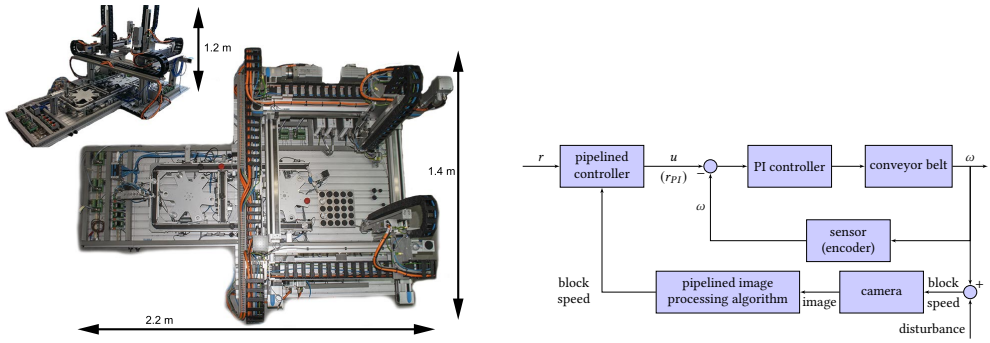

Fig. 7. Plant of feasibility study. (a) Assembly line xCPS. (b) Controller structure of one of the belts.

Table 2. Conveyor belt motor parameters.

Symbol	description	value	units
K_m	motor electrical constant	1.25	$^{vs}/_{rad}$
J_m	mechanical inertia	17.44	gm^2
R_m	motor resistance	10.39	Ω
b_m	viscous friction coefficient	0.58	Nms
K_p	controller proportional gain	4.10	
K_I	motor integral gain	166	

with $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} \omega & \int (r_{PI} - \omega) dt \end{bmatrix}^T$, ω the belt angular speed, and r_{PI} the reference sent to the PI controller. The model parameters are shown in Table 2. The resulting matrices are:

$$A_c = \begin{bmatrix} -70.36 & 1.14 \times 10^3 \\ -1 & 0 \end{bmatrix}, B_c = \begin{bmatrix} 28.31 \\ 1 \end{bmatrix}$$

The b_m presented in Table 2 corresponds to a nominal value. However, the actual value is estimated to vary $\Delta b_m = \pm 0.04$ around the nominal b_m. This range is caused because the motor is moving the belt whose friction is unknown but bounded to a range. The bounds on b are used to create the uncertain matrices:

$$\Delta A_c(t) = \begin{bmatrix} a(t) & 0 \\ 0 & 0 \end{bmatrix}, \Delta B_c(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

where

$$-\Delta b_m / J_m \leq a(t) \leq \Delta b_m / J_m$$
$$-2.30 \leq a(t) \leq 2.30$$

9.3 HIL simulation

9.3.1 Platform requirements. In order to make a HIL simulation for pipelined control, we need a platform with multi-core capabilities and a global notion of time. Multiple cores are required to independently run the controller in a pipelined fashion, as well as the plant model. A global notion of time is required to trigger the sensing and actuation tasks in each core at precise time instants, so that it guarantees sensing-to-actuating delay and the pipelined parallelism.

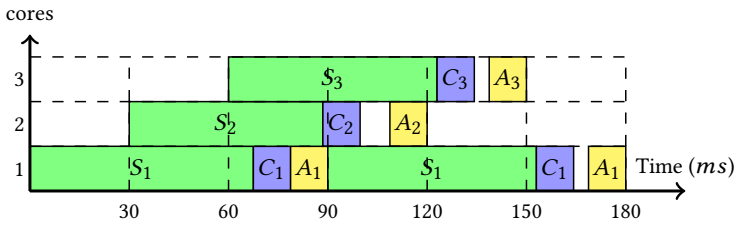

Fig. 8. Required platform for a three-cores pipelined controller for the system presented in Section 9.2. The colours follow those of Fig. 1.

Table 3. TDM slots for a three-core implementation on the CompSOC platform.

Table position	TDM position								
	1	2	3	4	5	6	7	8	9
core 1	S_1	S_1	S_1	S_1	S_1	S_1	S_1	C_1	A_1
core 2	S_2	C_2	A_2	S_2	S_2	S_2	S_2	S_2	S_2
core 3	S_3	S_3	S_3	S_3	C_3	A_3	S_3	S_3	S_3
	$90ms$								
	$30ms$								

Fig. 8 shows an example of the required platform for running a three-core pipelined controller in the plant of Section 9.2. The global notion of time allows to trigger a new sensing task every $30ms$. The total sensing to actuating latency corresponds to τ ($90ms$ in this case) in all the cores. Note that the image-processing algorithm latency may be shorter than its worst-case execution time. Therefore, the global notion of time is used to trigger the actuation tasks as close as possible to τ seconds after the start of the sensing task, achieving pipelined parallelism.

9.3.2 CompSOC platform. We use CompSOC, a multi-core tile-based architecture [14] as our implementation platform. In each core, the hardware resources are allocated according to a Time Division Multiplexing (TDM) table. The TDM tables are synchronized using a global clock, i.e. there is a global notion of time.

9.3.3 HIL set-up. Our HIL simulation uses the CompSOC platform with one core emulating the inner loop and one to three core tiles running the pipelined controller.

The core emulating the inner loop runs a discrete-time version of the model discretized at a sampling rate $h_p = 500\mu s$. h_p satisfies $h_p << h$, implying the plant appears to be continuous from the controller perspective.

In the cores emulating the pipelined controller the sensing, control computation, and actuation tasks are assigned to slots in the TDM tables. The task allocation has to guarantee that the time elapsed between the start of sensing and the end of actuating slots corresponds to τ. The actuation slot sends the controller input to the plant at the end of the actuation slot to guarantee τ. The task allocation across TDM tables also needs to guarantee pipelined parallelism, i.e. the start of the sensing task should be allocated such that pipelined parallelism is achieved. The global clock is then used to align the TDM tables among the cores.

The image-processing algorithm is not implemented in the HIL because of the lack of actual images. The latency of the image-processing algorithm is simulated by increasing the number of slots dedicated to the sensing task. The sensing information is read in the first sensing slot and delivered to the controller after the last sensing slot.

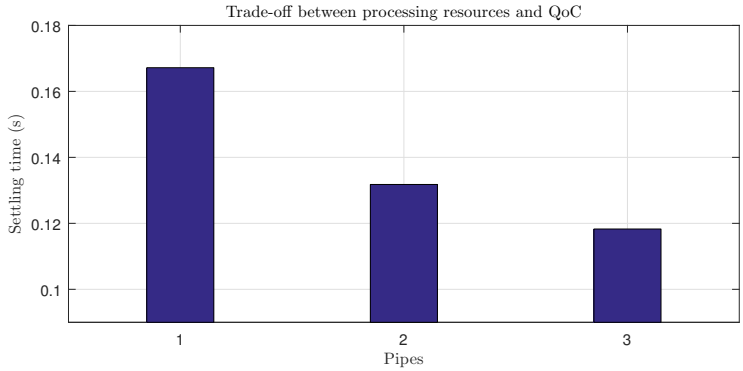

Fig. 9. Trade-off analysis without considering model uncertainties.

Table 3 shows an example of the TDM tables of a HIL simulation running three sensing pipes for the system of Section 9.2. The table length corresponds to τ guaranteeing the sensing-to-actuating latency. The TDM positions are synchronized among the three tables using the global clock. The tasks are allocated in such a way that a new sensing task starts every $30ms$, guaranteeing pipelined parallelism.

9.4 Application of the design flow

To find the resource configuration that improves the QoC while guaranteeing a robustness constraint. Following Example 7.6, we set the robustness constraint to $\alpha \geq 1$. We apply the design flow of Section 4 for the available core tiles $\gamma = 1 \ldots 3$. We show the analysis for the case of $\gamma = 3$.

Model discretization: Using the procedure explained in Section 3.2 we find a sampling period h and a discrete-time augmented model Φ_d, Γ_d for each resource configuration γ. For the resource configuration $\gamma = 3$, $h = 30ms$ and the model is given by:

$$\Phi_d = \begin{bmatrix} -0.0099 & 12.1327 & 0.5648 & 0 & 0 \\ -0.0106 & 0.7349 & 0.0203 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \Gamma_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Performance-oriented controller design: Using the procedure explained in Section 3.3, we compute a controller K, F for each resource configuration γ. For $\gamma = 3$, the controller is given by:

$$K = \begin{bmatrix} 11.24 & -2.47 & -102.26 & -188.94 & -432.09 \end{bmatrix} \times 10^{-3},$$
$$F = 1.803$$

The resulting trade-off between control performance and resource usage is shown in Fig. 9. In this case, each added core significantly improves the settling time. Therefore the optimal resource configuration (without considering the system robustness) is $\gamma = 3$.

Uncertain model benchmarking: The procedure explained in Section 6 is applied to the resource configurations. A pair of uncertainty matrices E, D is found for each resource configuration.

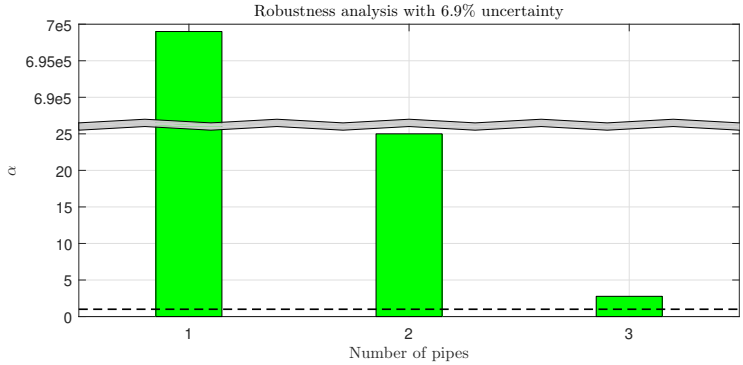

Fig. 10. Robustness analysis in the feasibility study

The resulting matrices for $\gamma = 3$ are:

$$E = \begin{bmatrix} 0.0037 & 0.2779 & 0 & 0 & 0 \\ -0.0002 & -0.0042 & 0 & 0 & 0 \end{bmatrix}, D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Robustness analysis for performance-oriented control: Applying Theorem 7.5 for the three considered resource configurations, an α for each resource configuration is found. For the case of $\gamma = 3$, $\alpha = 2.77$.

Trade-off analysis: Using the α of each resource configuration, Fig. 10 is drawn. Given that all the resource configurations have $\alpha \geq 1$, $\gamma = 3$ is chosen as the resource configuration that improves QoC while guaranteeing a robustness constraint.

9.5 HIL simulation results

We first validate our HIL simulation by comparing it with a Simulink-based simulation. Fig. 11 compares controller responses with three pipes. Both simulations show the same result implying that our HIL is correct and that pipelined control is feasible. Fig. 11 only shows information every h steps for the HIL simulation while for the Simulink simulation it shows a continuous line. This is as expected in a realistic scenario, because the only output information is available when a core finishes the image-processing algorithm, i.e., every h seconds.

Fig. 12 compares the controller performance using the three available resource configurations. Every newly added pipe improves the QoC of the system. The resource configuration with three pipes outperforms the other two, due its higher sampling rate.

The robustness of the pipelined controller is also tested using HIL simulation. We add to the nominal model a constant uncertainty of $a = 1.15$. The model is discretized and implemented in the corresponding core tile. The result is shown in Fig. 13 for $\gamma = 3$. The controller is still stable, although the settling time is increased from $S_t = 118ms$ in the nominal case to $St = 138ms$ in the uncertain case.

10 CONCLUSIONS AND FUTURE WORK

We have presented a method to either analyse the impact of uncertain models in performance-oriented pipelined controllers or to design robustness-oriented pipelined controllers. Our method

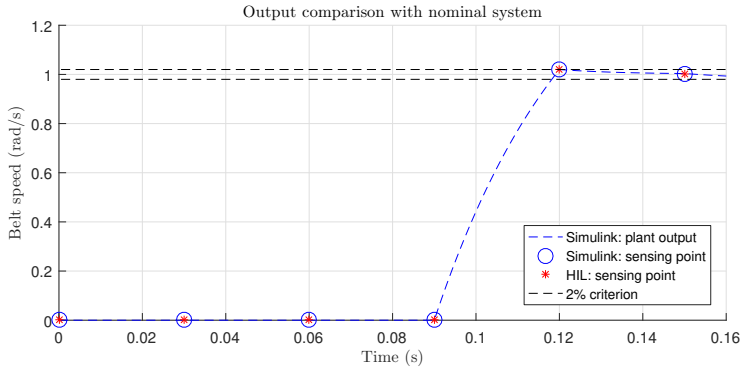

Fig. 11. Performance comparison 3 cores pipelined controller.

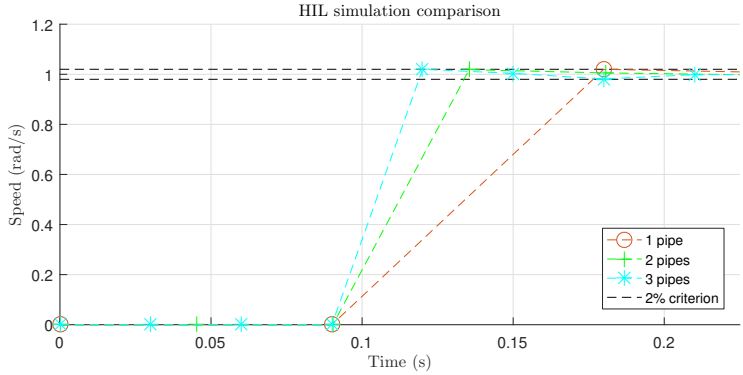

Fig. 12. Step responses comparison with different resource configurations using HIL simulation.

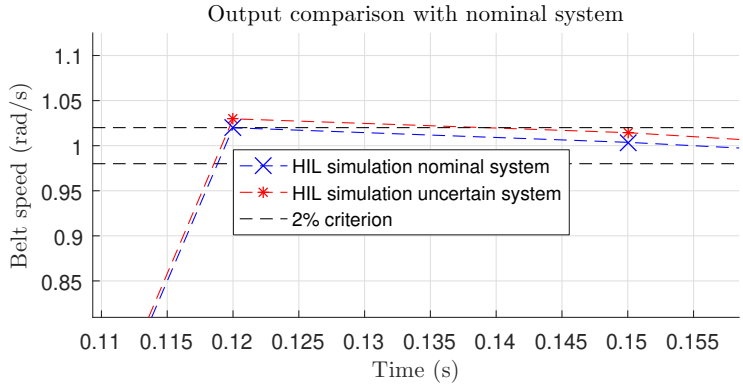

Fig. 13. Output of HIL simulation when the uncertain system is considered.

starts with a technique to benchmark discrete-time uncertainties based on continuous-time uncertainties. Such uncertainties are presented in the form of norm-bounded time-variant additive matrices with one uncertain element. Our results show that the robustness of a performance-oriented pipelined controller decreases with each newly added pipe, because adding sensing resources

produces a more aggressive response which is more susceptible to modelling errors. Therefore, our analysis may be used in a performance-oriented design to select a resource configuration that not only provides performance improvement but also guarantees that a minimum desired robustness is met. Likewise, our results show that the robustness of pipelined controller can be increased with each newly added pipe in a robustness-oriented design. Therefore, our method can be used to select a resource configuration that provides a desired robustness. We presented a Hardware-In-the-Loop (HIL) simulation to show the feasibility of pipelined control. The HIL simulation validates our analysis, results, and observations.

As future work, we plan to consider variable sensing delays instead of abstracting them as the worst-case delay (which may occur rarely). Further, it would be interesting to develop an analysis for a more general class of uncertainties. Finally, existing design methods for pipelined control may provide insight whether it is possible to design a controller that improves robustness and control performance at the same time.

REFERENCES

[1] S. Adyanthaya et al. 2017. xCPS: A Tool to Explore Cyber Physical Systems. *SIGBED Rev.* 14, 1 (2017), 81–95.

[2] R. Agrawal et al. 2014. A GPU based Real-Time CUDA implementation for obtaining Visual Saliency. In *Proc. ICVGIP*. ACM.

[3] M. Bacic. 2005. On hardware-in-the-loop simulation. In *Proc. of the 44th IEEE Conference on Decision and Control*. 3194–3198.

[4] M. Braga, C. Morais, L. Maccari, E. Tognetti, V. Montagner, R. Oliveira, and P. Peres. 2017. Robust Stability Analysis of Grid-Connected Converters Based on Parameter-Dependent Lyapunov Functions. *Journal of Control, Automation and Electrical Systems* 28, 2 (2017), 159–170.

[5] M. F. Braga, C. F. Morais, E. S. Tognetti, R. C. L. F. Oliveira, and P. L. D. Peres. 2013. A new procedure for discretization and state feedback control of uncertain linear systems. In *Proc. CDC*. IEEE, 6397–6402.

[6] F. Chaumette and S. Hutchinson. 2006. Visual servo control. I. Basic approaches. *IEEE Robotics and Automation Magazine* 13 (2006), 82–90.

[7] S. Chroust et al. 2000. Evaluation of processing architecture and control law on the performance of vision-based control systems. In *Proc. AMC*. IEEE, 19–24.

[8] Jamal Daafouz, Pierre Riedinger, and Claude Iung. 2002. Stability analysis and control synthesis for switched systems: a switched Lyapunov function approach. *IEEE transactions on automatic control* 47, 11 (2002), 1883–1887.

[9] D. Du, B. Jiang, and S. Zhou. 2008. Delay-dependent robust stabilisation of uncertain discrete-time switched systems with time-varying state delay. *International Journal of Systems Science* 39, 3 (2008), 305–313.

[10] H. Gao and T. Chen. 2007. New Results on Stability of Discrete-Time Systems With Time-Varying State Delay. *IEEE Trans. on Automatic Control* 52, 2 (2007), 328–334.

[11] P. Garcia, P. Castillo, R. Lozano, and P. Albertos. 2006. Robustness with respect to delay uncertainties of a predictor-observer based discrete-time controller. In *Proc. CDC*. IEEE, 199–204.

[12] J. Geromel, M. de Oliveira, and J. Bernussou. 2002. Robust Filtering of Discrete-Time Linear Systems with Parameter Dependent Lyapunov Functions. *SIAM J. Control Optim.* 41, 3 (2002), 700–711.

[13] A. Gonzalez, A. Sala, P. Garcia, and P. Albertos. 2013. Robustness analysis of discrete predictor-based controllers for input-delay systems. *International Journal of Systems Science* 44, 2 (2013), 232–239.

[14] K. Goossens et al. 2017. *NoC-Based Multiprocessor Architecture for Mixed-Time-Criticality Applications*. Springer Netherlands.

[15] L. Hetel, J. Daafouz, and C. Iung. 2007. LMI control design for a class of exponential uncertain systems with application to network controlled switched systems. In *Proc. ACC*. 1401–1406.

[16] R. Isermann, J. Schaffnit, and S. Sinsel. 1999. Hardware-in-the-loop simulation for the design and testing of engine-control systems. *Control Engineering Practice* 7, 5 (1999), 643 – 653.

[17] K. Jo et al. 2014. Development of Autonomous Car—Part I: Distributed System Architecture and Development Process. *Industrial Electronics, IEEE Trans. on* 61, 12 (2014), 7131–7140.

[18] K. Jo et al. 2015. Development of Autonomous Car—Part II: A Case Study on the Implementation of an Autonomous Driving System Based on Distributed Architecture. *Industrial Electronics, IEEE Trans. on* 62, 8 (2015), 5119–5132.

[19] S. Kestur et al. 2012. Emulating mammalian vision on reconfigurable hardware. In *Proc. FCCM*. IEEE, 141–148.

[20] P. Khargonekar, I. Petersen, and K. Zhou. 1990. Robust stabilization of uncertain linear systems: quadratic stabilizability and H ∞ control theory. *IEEE Trans. on Automatic Control* 35, 3 (1990), 356–361.

[21] M. Kothare, V. Balakrishnan, and M. Morari. 1996. Robust constrained model predictive control using linear matrix inequalities. *Automatica* 32, 10 (1996), 1361 – 1379.

[22] P. Krautgartner and M. Vincze. 1998. Performance evaluation of vision-based control tasks. In *Proc. of ICRA*, Vol. 3. IEEE, 2315–2320.
[23] Peter Krautgartner and Markus Vincze. 1999. Optimal Image Processing Architecture for Active Vision Systems. In *Computer Vision Systems*. Lecture Notes in Computer Science, Vol. 1542. Springer, 331–347.
[24] M. A. C. Leandro, J. R. C. Júnior, and K. H. Kienitz. 2015. Robust D-stability via discrete controllers for continuous time uncertain systems using LMIs with a scalar parameter. In *Proc. MED*. IEEE, 644–649.
[25] F. Lewis and V. Syrmos. 1995. *Optimal control*. John Wiley & Sons.
[26] J. Löfberg. 2004. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *Proc. CACSD*. Taipei, Taiwan.
[27] M. Loper. 2015. *Modeling and simulation in the systems engineering life cycle: core concepts and accompanying lectures*. Springer.
[28] R. Lozano, P. Castillo, P. Garcia, and A. Dzul. 2004. Robust prediction-based control for unstable delay systems: Application to the yaw control of a mini-helicopter. *Automatica* 40, 4 (2004), 603 – 612.
[29] R. Medina, S. Stuijk, D. Goswami, and T. Basten. 2016. Reconfigurable pipelined sensing for image-based control. In *Proc. SIES*. IEEE.
[30] R. Medina, S. Stuijk, D. Goswami, and T. Basten. 2017. Exploring the trade-off between processing resources and settling time in image-based control through LQR tuning. In *Proc. SIGAPP SAC*. ACM.
[31] K. Åström and B. Wittenmark. 1997. *Computer-controlled Systems (3rd Ed.)*. Prentice-Hall, Inc.
[32] W. Rugh. 1996. *Linear system theory*. Vol. 2. Prentice hall Upper Saddle River, NJ.
[33] P. Sharkey and D. Murray. 1996. Delays versus performance of visually guided systems. *Control Theory and Applications, IEE Proc.* 143, 5 (1996), 436–447.
[34] L. Shieh, W. Wang, and G. Chen. 1998. Discretization of cascaded continuous-time controllers and uncertain systems. *Circuits, Systems and Signal Processing* 17, 5 (1998), 591–611.
[35] K. Toh, M. Todd, and R. Tütüncü. 1999. SDPT3, a Matlab software package for semidefinite programming, Version 1.3. *Optimization Methods and Software* 11, 1-4 (1999), 545–581.
[36] G. van den Braak et al. 2011. Fast Hough Transform on GPUs: Exploration of Algorithm Trade-Offs. In *Advanced Concepts for Intelligent Vision Systems*. Vol. 6915. Springer, 611–622.
[37] N. Wada, K. Saito, and M. Saeki. 2004. Model predictive control for linear parameter varying systems using parameter dependent Lyapunov function. In *Proc. MWSCAS*, Vol. 3. iii–133–6 vol.3.
[38] F. Xia, Z. Wang, and Y. Sun. 2004. Allocating IEC function blocks for parallel real-time distributed control system. In *Proc. CCA*, Vol. 1. IEEE, 254–259.
[39] F Xia, Z. Wang, and Y. Sun. 2004. Design and evaluation of event-driven networked real-time control systems with IEC function blocks. In *Proc. SMC*, Vol. 6. IEEE, 5148–5153 vol.6.
[40] Y. Xia, G. Liu, P. Shi, D. Rees, and E. Thomas. 2007. New Stability and Stabilization Conditions for Systems with Time-delay. *International Journal of System Sciencie* 38, 1 (2007), 17–24.
[41] L. Xie. 1996. Output feedback H ∞ control of systems with parameter uncertainty. *International Journal of control* 63, 4 (1996), 741–750.
[42] L. Xie, M. Fu, and C. de Souza. 1992. H ∞ control and quadratic stabilization of systems with parameter uncertainty via output feedback. *IEEE Trans. on Automatic Control* 37, 8 (1992), 1253–1256.
[43] L. Xie, M. Fu, and C. de Souza. 1992. H $_\infty$ control and quadratic stabilization of systems with parameter uncertainty via output feedback. *IEEE Trans. on Automatic Control* 37, 8 (1992), 1253–1256.
[44] L. Yao et al. 2009. An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher. In *Proc. FPT*. IEEE, 30–37.
[45] M. Yuan, Z. Wang, X. Ren, H. Yu, and Y. Zhou. 2003. Function block-based pipelined controller. In *Proc. IECON*. IEEE, 1956–1961.
[46] F. Zhang. 2006. *The Schur complement and its applications*. Vol. 4. Springer Science & Business Media.
[47] K. Zhou and J. Doyle. 1998. *Essentials of robust control*. Prentice hall Upper Saddle River.
[48] G. Zong, L. Hou, and H. Yang. 2009. Further results concerning delay-dependent control for uncertain discrete-time systems with time-varying delay. *Mathematical Problems in Engineering* (2009).

Received December 2017; revised March 2018; accepted June 2018